ILT1000, 2400, 2500, 5000 API Documentation

Revision Table

Date	Revision	Comment
May 21 2019	V2.7	• Major release with several updates. • echoon (verbose) mode has been deprecated in firmware release 3.2.2.7 and later.

ILT API Documentation

The ILT device uses a simple text-based API for communication with user applications, as well as for use with a basic terminal emulation software.

Document Convention

The API table below contains two columns. In the first column is the API/command name. In the second column is the API/command's definition. The definition contains several elements and convention as follows.

- Actual command and responses will use the `Courier New` font.
- Command parameters are identified in [brackets], within the "Syntax" description. *The brackets themselves are not used in the command syntax*, as is illustrated by the "Example command" listed for some of the commands.
- An N/A for an "Error return value(s)" indicates that errors are not returned and not expected for this command.
- "Example return" is listed, on some commands, to provide an example of the output. In this field, any text following a dash ("-"), as well as the dash itself, are not part of the actual output and only listed to help describe the output.
- Commands are listed alphabetically within each section.
- The optional "Persist through power-cycle" field will indicate whether or not the configuration is stored in flash memory and "sticks" through power cycles.
- All commands are case sensitive, and always all-lowercase.
- The "Support Starting in Firmware Version" data indicates the firmware version where the command is supported by the manufacturer. The actual command may be present in earlier versions of firmware, but not officially supported.

Theory of Operation

Below is the Theory of Operation for interfacing with the device using the API defined further below.

NOTE: Firmware version 3.1.4.7 or later is recommended for the best API response times.

- The device responds to a number of commands (together known as the command-line API) delivered via the USB port (most commonly), UDP, RS485, or some custom serial interface.
- The USB port is configured on the device as a USB Serial Port. As a result, interfacing can be done with any standard terminal programs (hyperterminal, puTTY, MAC terminal window, etc), or direct serial port programming via C, C#, LabView, MatLab, etc.
    - Serial port settings are:
        - Baud Rate: 115200
        - Data Bits: 8
        - Parity bits = None
        - Stop bits = 1
        - Flow Control = None
- The device behaves as follows and in sequence:
    a. Perform analog/digital conversion and mathematical functions as part of optical level detection
        i. Buffer up to 4 characters of any incoming commands while performing the processing above
    b. Check for any characters in the buffer referenced above to indicate an incoming command and process any commands as required. Important programming notes:
        i. Starting in firmware version 3.1.4.7, it will typically take less than 10ms to complete complex analog/digital tasks and process the remainder of the command that was not buffered. Earlier firmware versions can take as long as 50ms.
        ii. The device determines the completion of the command by detecting a "\r" character, i.e. ASCII code 13 decimal.
    c. Start all over from the top at (a)….
- As a result of the above device behavior, the recommended method to program each device is to:
    - Always append commands with \r to indicate the command completion.
        - Do NOT send a "\r\n" command as the "\n" will be interpreted as the start of the next command, typically resulting in a delay followed by a "-999\r\n" response indicating an unknown command. Some programming languages have API's that automatically attach a termination character. For example C# SerialPort.WriteLine(…) will append the "\n" by default. In this case use SerialPort.Write(…) with the "\r" embedded in the string passed to the API.
        - Note that "\r" is not two characters, as would be typed at a terminal or sent as part of a string by some programming API's. It represents the ASCII code 13 or a carriage return.
    - Send the first character of the command, for example the "g" in "getcurrent\r".
        - It is good practice to "drain" the serial line input prior to sending any commands. This involves simply reading any characters that might be stuck in the serial input buffer prior to sending a new command.
    - Pause 10ms
        - For firmware versions earlier than 3.1.4.7, this needs to be extended to 50ms.
    - Send the remainder of the command, i.e. "etcurrent\r".
        - IMPORTANT: Apple/MAC and some Unix-based systems have shown a need for a 1ms inter-character delay.
    - Immediately start sensing the response from the device.
        - The device will always send a response to acknowledge the status of the command completion as well as to return values for "get" commands.

        - It is important to always read the response back, even if there is no interest in the response, to empty the serial input buffer.
    - The response will always be terminated with a "\r\n" (13 decimal, 10 decimal) sequence. This can be used to sense the end of the response and start sending the next command.
    - Be careful not to expect an immediate response from the device. For instance, after sending the remainder of the command, i.e. "etcurrent\r", do not immediately check the receive buffer and give-up if data is not sensed. It can take over 1 ms for even the fastest commands to return. It is better (more robust) to (a) send the remainder of the command, (b) continue looking for and processing a response until "\r\n" is sensed, and (c) use a timeout to detect when a response has not returned within some time-out period.
        - Regarding time-outs for command responses, "get" commands will typically respond within 100ms. "set" commands that store their configuration in flash memory can take up to 5 seconds to respond. Other special commands like "setuserdark" and "captureflash" can take longer to respond.
    - There are certain commands that will return multiple lines, i.e. multiple "\r\n" terminations in response to a single command.
    - For commands that return a large amount of data, such as getlogdata, ensure that the receiving buffer is large enough to hold all the data.
- Each device is single threaded, meaning that commands and responses need to be processed in sequence. A 2nd command cannot, for example, be initiated before the 1st command's response is fully processed. As a result, if a multi-threaded application is accessing the device, a programmatic lock must be placed around device command/response sequences to make sure multiple threads do not attempt to access the device at the same time.
    - If multiple devices are being monitored, a single lock can be used for access to all devices. This is simpler from a coding perspective, but it is not as efficient as it does not allow multiple devices to operate in parallel. For the best performance it is recommended that a per-device lock be established. This allows all devices to be accessed in parallel.
    - As a further performance benefit when monitoring multiple devices, the delay after the first character can be performed in parallel across all devices. For example, if sending "getcurrent\r" to 5 devices, one would:
        - Send "g" to all 5 devices
        - Wait 10ms
        - Send "etcurrent\r" to all devices
        - Process the reply from all devices
            - The replies can be processed in "round-robin" fashion, allowing commands that return a large amount of data, such as getlogdata, to be processed faster.
- Many of the more popular command have had 2-character short-cuts added over time. This allows rapid, repeated access to sensor data because the command fits in the 4-character buffer and does not require a delay after the first character of the command. These commands, and short-cuts, are as follows:
    - gc = getcurrent, introduced in FW version 3.0.5.4
    - gi = getirradiance, introduced in FW version 3.0.5.4
    - gv = getvoltage, introduced in FW version 3.0.5.4
    - gt = gettrans, introduced in FW version 3.0.9.4
    - go = getod, introduced in FW version 3.0.9.4

API

`captureflash`

Support Starting in Firmware Version: 3.0.5.8

Support Ending in Firmware Version: N/A

Syntax:

```
captureflash [trigger type] [trigger level] [minimum light level] [timeout period] [voltage sensitivity] [trigger offset time] [integration time] [save to flash] [save only above minimum light level]
```

What it does:

This command starts a rapid sampling of light data, intended for capturing "flash" sources such as camera flashes, safety flashing lights, etc.

trigger type:
0 = trigger in (start capturing "trigger offset time" milliseconds before trigger in signal)
1 = trigger out (start capturing "trigger offset time" milliseconds after trigger out signal)
2 = trigger on light level (start capturing after "minimum light level" exceeded)
3 = manual trigger (start capturing immediately)

trigger level (applies to trigger type 0 and 1, ignored otherwise):
low = trigger on logic low condition (transition from high to low)
high = trigger on logic high condition (transition from low to high)

min light level
Floating point value representing the minimum light level above which capture will start when trigger type is set to 2. Ignored (enter 0.00) otherwise.

timeout period
The number of seconds (0 to 300) that the routine will wait for a trigger. Ignored for manual trigger.

voltage sensitivity
1 = lowest sensitivity voltage gain (brightest light)
2 = medium sensitivity voltage gain
3 = highest sensitivity voltage (lowest light level)

trigger offset time
The number of milliseconds (0-20) that the trigger will be offset as follows, by trigger type:
0 : not applicable
1 : number of milliseconds before trigger in
2 : number of milliseconds after trigger out
3 : number of milliseconds before minimum light level is sensed

integration time
The number of milliseconds (1-40000) that the flash signal will be captured

save to flash
0 = do not save to flash (just capture results, see `getflash`)
1 = save to flash (4096 data points are saved across the integration time, use `getlogdata` to download)

save only above minimum light level (applicable to trigger type 2)
0 = save all data points
1 = only save data points above the minimum light level

Normal return value:
0 on success

`captureflash` (Continued)	Error return value(s): -500 if missing fields -501 if bad trigger type. Must be 0-3. -502 if bad trigger level. Must be "low" or "high" -503 if bad minimum light level. Must be greater than 0. -504 if bad timeout period. Must be 0-300. -505 if bad voltage sensitivity. Must be 1-3. -506 if bad trigger offset. Must be 0-20. -507 if bad integration time. Must be 1 – 40000. -508 if bad save to flash value. Must be 0 or 1. -509 if bad save only above min value. Must be 0 or 1. -510 if calibration factor not in use. See `usecalfactor`. -511 if save to flash is set to 1, but data is already in flash. See `eraselogdata`. -512 if timeout period expired before trigger was completed -513* if a minimum light level is too high to be measured (device will saturate before reaching the level) with given range (Rf + voltage sensitivity) *Added in 3.0.6.7 Persist through power-cycle: N/A Example command (trigger on light level, "low" trigger level ignored, minimum light level = 20e-3 calibrated units, 2 second timeout, Low voltage sensitivity, 5 millisecond trigger offset, 40 millisecond integration time after trigger, save data to flash for later retrieval, save only the light levels above the minimum trigger level): `captureflash 2 low 20e-3 2 1 5 40 1 1`
`clearambientlevel` Support Starting in Firmware Version: 3.0.5.8 Support Ending in Firmware Version: N/A	Syntax: `clearambientlevel` What it does: This command removes any ambient levels configured with `setambientlevel`. Normal return value(s): 0 on success Error return value(s): N/A Persist through power-cycle: No

`erasecalfactor`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`erasecalfactor [calibration number, 1-20]`
	
	What it does:
	This command erases the calibration data associated with the calibration factor. This includes the calibration factor description, the current-to-irradiance multiplier, and the saturation current.
	
	Starting in 3.0.5.3, if the active calibration factor is erased, the cal factor is set to 0 indicating no cal factor in use.
	
	Normal return value(s):
	0 on success
	
	Error return value(s):
	-500 if missing fields
	-501 if bad factor number
	-502 if error erasing flash
	
	Persist through power-cycle:
	Yes
	
	Example command:
	`erasecalfactor 5`

`eraselogdata` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `eraselogdata` What it does: This command erases any log data that has been stored in the device's flash memory. See also `startlogdata` and `stoplogdata`. Normal return value(s): 0 on success Error return value(s): -500 if logging is currently active (must use stoplogdata first) -501 error erasing the data from flash Persist through power-cycle: Yes

`get100perc` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `get100perc` What it does: Prior to 3.0.5.3, this command returns the sensor voltage at the time the 100% value was set with `set100perc`. Starting in 3.0.5.3, this command returns the sensor current at the time the 100% value was set with `set100perc`. Normal return value(s): Prior to 3.0.5.3: Voltage for 100% transmission, volts (firmware >= 2.1.0.0) or microvolts (firmware<2.1.0.0) Starting in 3.0.5.3: Current for 100% transmission. Error return value(s): -500 if the 100% value was never set

`getambientlevel`
 	
Support Starting in Firmware Version: 3.0.5.8
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getambientlevel`
	
	What it does:
	This command returns the existing ambient light level that is removed from all current and light level readings.
	
	Normal return value(s):
	Ambient light level, based on detector current (units = amps)
	
	Error return value(s):
	N/A
	
	Persist through power-cycle:
	No
	
	Example return:
	`5.981e-6`

`getambienttemp (Gen2, Gen3)` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `getambienttemp` What it does: This command returns the ambient temperature as sensed by the device. This differs from `gettemp` which returns the internal temperature of the microcontroller. Normal return value(s): Temperature in degrees F (x100 for firmware < 2.1.0.0). Error return value(s): -500 Command not supported

`getapiversion` Support Starting in Firmware Version: 2.1.0.0 Support Ending in Firmware Version: N/A	Syntax: `getapiversion` What it does: This command returns the api version for the firmware. From time-to-time, the CLI API return values may change the return value or formats. When this happens the API version is increased such that programs can proactively determine the expected return values. API Changes from v1 to v2: • get100perc, getvoltage, getvx1, getvx17, getvagc3, getvped, getvref, set100perc all changed from returning microvolts to returning volts in decimal form • getcurrent changed from returning picoamps to amps in scientific notation • getirradiance changed from returning "pico" values to standard values in scientific notation • gettrans changed from returning percent transmission x 10 to percent transmission in decimal form • getod changed from returning optical density x 100 to optical density in decimal form • getambienttemp changed from returning temp x 100 to tempx1 • getlogdata return values changed to reflect unit changes above, with all values returned in scientific notation API Changes from V2 to V3 • Added new API's, noted in this document as Supported Starting in Firmware Version 3.0.5.3 or later. • set100perc and get100perc changed to return current as opposed to voltage • deprecated setclockfreq, setirrdatapoint, storeirrdata, eraseirrdata, setsimpleirrcal, hiddenhelp • deprecated setsamplecount (use setsampletime) and getsamplecount (see getsampletime) • Overloaded usefeedbackres to return the setting in use if no parameters are provided Normal return value(s): -999 for firmware versions earlier than 2.1.0.0, indicating API version 1. 2 for firmware versions 2.1.0.0 and greater 3 for firmware versions 3.0.5.3 and greater Error return value(s): N/A

`getauxserialno`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getauxserialno`
	
	What it does:
	This command returns the auxiliary serial number of the device. The serial number is stored in one-time-programmable memory at the time of manufacture.
	
	Normal return value(s):
	Device serial number
	
	Error return value(s):
	-500 device serial number has not been set
	
	Example return:
	`sn17839-0001`

`getbias` Support Starting in Firmware Version: 3.0.5.4 Support Ending in Firmware Version: N/A	Syntax: `getbias` What it does: This command returns the level of bias voltage applied to the detector. Normal return value(s): 0 indicating no bias is applied 5 indicating that voltage is applied such that the anode is 5V more negative than the cathode Error return value(s): -501 if not supported on the device

`getcalfactor`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getcalfactor [optional, calibration factor, 1-20]`
	
	What it does:
	Without the optional parameter, this command returns the calibration factor currently in use. With the optional parameter, this command returns the calibration factor details for a particular calibration factor. This includes the following information, separated by spaces: calibration factor description with optional units, current-to-irradiance divisor, and saturation current in microamps. A typical use case is to use the command without the optional parameter to determine which calibration factor is in use, followed by issuing the command with the optional calibration factor to determine the details of the calibration factor definition.
	
	Normal return value(s):
	Without the optional command line parameter:
	
	0 if no calibration factor in use
	1-20 indicating which calibration factor is in use
	
	With the optional command line parameter:
	
	[calibration factor description] [current-to-irradiance multiplier(x1000)] [saturation current]
	
	Error return value(s):
	-501 if calibration factor is out of the 1-20 range
	-502 if the calibration factor is not defined
	
	Example command:
	`getcalfactor 1`
	
	Example return (note the optional units specifier ‘W’ after colon):
	`calfact1:W 2.7e-6 50`

`getcurrent` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `getcurrent` Shortcut: `gc` (support for this shortcut starting in 3.0.5.4) What it does: This command returns the sensor current in picoamps. Normal return value(s): Current in amps (firmware>=2.1.0.0) or picoamps (firmware<2.1.0.0) Error return value(s): -500 if there is a voltage saturation or, starting in FW 3.1.3.4 a diode saturation per the cal factor setting.
`getdarkmode` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `getdarkmode` What it does: This command returns the dark mode currently in use by the device. The dark mode can be NO DARK (there is no consideration for photodiode dark current), FACTORY DARK (the dark current set at the factory), or USER DARK (the dark current set with `setuserdark`). Return value(s): 0 = NO DARK 1 = FACTORY SET 2 = USER SET Error return value(s): N/A

`getdatetime (Gen2, Gen3)`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getdatetime`
	
	What it does:
	This command returns the real time clock's date and time in formats as shown below. This format includes a typical date and time format (mm/dd/yyyy hh:mm:ss), followed by the Epoch time (seconds since 1970). Note that the device is programmed to UTC/GMT time and, as a result, might return a time that does not match the local timezone. Modern computer programming languages will correctly convert the Epoch time data to local time when saving the timestamped log data. This is the case with the Data Logger software.
	
	Return value(s):
	Real time clock time as shown below
	
	Error return value(s):
	-500 Command not supported
	
	Example return:
	`12/05/2013 19:02:05 1386270125`

`getecaldate (Gen3)` Support Starting in Firmware Version: 3.0.5.3 Support Ending in Firmware Version: N/A	Syntax: `getecaldate` What it does: This command returns the sensor last date electrical cal (ecal) was performed, in epoch time. If an ecal as never performed, this api returns 0. Normal return value(s): 0 if ecal was never performed Date in epoch time if ecal was performed Example return: 1428445793 (indicating April 7 2015 22:29:53 GMT) Error return value(s): N/A

`getfactorydark`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getfactorydark`
	
	What it does:
	This command returns the dark voltage at the various transimpedance amplifier gain stages, as set at the factory.
	
	Normal return value(s):
	On Gen1 products: The dark voltage in microvolts, of the two voltage gain stages.
	On Gen2 products: The dark voltage in microvolts, of the three voltage gain stages for each of three feedback resistor stages
	On Gen3 products: The dark voltage in microvolts, of the three voltage gain stages for each of four feedback resistor stages
	
	Error return value(s):
	-500 if the factory dark voltage has not been set
	
	Example return:
	`Gen1:`
	`12756 9234`
	`Gen2:`
	`R1 10360 9602 9535 R2 14115 13291 13215 R3 46680 45769 25190`

`getfeedbackresnumber` Support Starting in Firmware Version: 3.0.5.8 Support Ending in Firmware Version: N/A	Syntax: `getfeedbackresnumber` What it does: This command returns the number, not the value, of the feedback resistor in use. Normal return value(s): 1 for Gen1 devices (only 1 feedback resistor) 1 – 3 for Gen2 devices 1 – 4 for Gen3 devices Error return value(s): N/A

`getflash`
 	
Support Starting in Firmware Version: 3.0.5.8
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getflash`
	
	What it does:
	This command returns the results of the captureflash command, including the following data:
	
	Peak: This is the peak light level sensed during the flash capture
	Average: This is the average light level sensed over the integration period
	Integration: This is the integration of the light level over the integration period
	Time Above 10 Percent of Peak: This is the time (in seconds) that the signal was above 10% of the peak value. This can be considered the duration of the pulse.
	Peak Percent of Range: This is the percentage of the input range (defined by the feedback resistor in use and the voltage sensitivity). A number above 95 would indicate a likely saturation of the signal.
	
	The data labels, equal signs, and actual results are all separated by a space, facilitating parsing with a String->Split(' ') command.
	
	Normal return value(s):
	See example return below.
	
	Error return value(s):
	The 'Peak' value is returned as -512 if the capture resulted in a timeout before the trigger was sensed. In the response below, the Peak = -5.120e+02 indicates this condition, making the remainder of the values are invalid.
	
	`Peak = -5.120e+02 Average = 6.168e-06 Integral = 2.527e-07 Time-Above-10-Percent-of-Peak = 4.096e-02 Peak-Percent-of-Range = 0`
	
	Example return:
	`Peak = 1.067e-03 Average = 1.301e-05 Integral = 1.041e-06 Time-Above-10-Percent-of-Peak = 7.031e-04 Peak-Percent-of-Range = 97`

`getfeedbackres`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getfeedbackres`
	
	What it does:
	This command returns the value of the feedback resistor, in kilo-Ohms x 10.
	
	Normal return value(s):
	Transimpedance amplifier feedback resistance in kOhms x 10
	
	Error return value(s):
	N/A
	
	Example return (for a 3K Ohm feedback resistor):
	`30`

`getfriendlyname` Support Starting in Firmware Version: 3.0.5.4 Support Ending in Firmware Version: N/A	Syntax: `getfriendlyname` What it does: This command returns the device's "friendly name". Normal return value(s): Device friendly name or "NOT-DEFINED" if the friendly name has not yet been saved Error return value(s): N/A

`getfwversion`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getfwversion`
	
	What it does:
	This command returns the firmware version running on the device.
	
	Normal return value(s):
	Device firmware version
	
	Error return value(s):
	N/A
	
	Example return:
	`1.3.0.5`

`getgeneration`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getgeneration`
	
	What it does:
	This command returns the generation of the device.
	
	Normal return value(s):
	-999 for first generation products that did not have this command defined at time of release
	1 for first generation products
	2 for second generation products (programmable Rf x 3, realtime clock, ambient temp sensor)
	3 for second generation products (programmable Rf x 4, realtime clock, ambient temp sensor)
	
	Error return value(s):
	N/A
	
	Example return:
	`2`

`getinfo`
 	
Support Starting in Firmware Version: 3.0.5.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getinfo`
	
	What it does:
	This command returns a [growing] list of critical device parameters. The command is intended as a quick diagnostic and configuration check, but can be used with parsing software to capture and extract critical configuration information.
	
	Normal return value(s):
	See command output for the particular firmware version.
	
	Error return value(s):
	N/A
	
	Example return:
	`Base Serial Number = 10002201407300019`
	`Vendor Serial Number = ILT100000002`
	`Model Name = ILT1000-V02`
	`Friendly Name = Right`
	`Generation = 2`
	`FW Version = 3.2.2.7`
	`Dark mode (0=No,1=Factory,2=User) = 1`
	`Rf Setting (0=Auto,1=Rf1,2=Rf2,3=Rf3,4=Rf4) = 0`
	`Rf value R1 (kOhms) = 3`
	`Rf value R2 (kOhms) = 1000`
	`Rf value R3 (kOhms) = 10000`
	`Rf value R4 (kOhms) = 10000000`
	`Rf value in use (KOhms) = 10000`
	`eCal: Disabled`
	`eCal values = 1.000e+00 1.000e+00 1.000e+00 1.000e+00`
	`1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00`
	`1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00`
	`1.000e+00 1.000e+00 1.000e+00`
	`eCal Temp (F) = 0.00`
	`Sample time (ms) = 500`
	`Auto Sample Time: Enabled`
	`4-20mA mode = 8`
	`Current Loop Min Max: 0.000e+00 1.000e+03`
	`getvx1 = 0.019336`
	`getvx17 = 0.300513`
	`getvagc3 = 2.344482`
	`Active voltage gain stage = 3`
	`TIA voltage = -0.005088`
	`getcurrent = 0.000e+00`
	`getirradiance = 0.000e+00`
	`Calibration factor = 1.000e+00`
	`Factory dark = R1 9627 9830 9893 R2 10844 11230 11286 R3 22550`
	`22904 22985`
	`User dark = -500`
	`Ambient level = 0.000e+00`
	`Logging: Disabled`
	`Wireless Listening: Disabled`
	`Peak Tracking: Disabled`
	`Fast Integrate: Enabled`
	`Multidrop: Disabled`

`getintegrate`
 	
Support Starting in Firmware Version: 3.0.7.0
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getintegrate`
	`getintegrate nolowerror (starting in FW 3.2.2.7)`
	
	What it does:
	This command returns the integrated light level that started with `startintegrate`.
	
	Starting in firmware release 3.2.2.7, the `nolowerror` qualifier will result in the -501 error (see below) being ignored. This is used for cases where it has been determined that the light level can be integrated with enough accuracy with feedback resistor 1.
	
	Normal return value(s):
	Integrated light level (units depend on the calibration factor)
	
	Error return value(s):
	-500 if a diode saturation or gain stage voltage saturation condition was sensed
	-501 special case for instance where feedback resistor is 1 (3K), and peak current < 3uA
	
	Example return (for an integrated light level value of 4.712 milli-units):
	`4.712-3`

`getirradiance`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getirradiance`
	
	Shortcut:
	`gi` (support for this shortcut starting in 3.0.5.4)
	
	What it does:
	This command returns the irradiance value in user-defined units.
	
	Normal return value(s):
	Irradiance value in user-defined units
	
	Error return value(s):
	-500 if no irradiance calibration factor is in use
	-502 if there is a voltage saturation or, starting in FW 3.1.3.4 a diode saturation per the cal factor setting, indicating the reading must be discarded
	
	Example return (for an irradiance value of 7.798 milli-units):
	`7.798e-3`

`getirrthresholdlow`
 	
Support Starting in Firmware Version: 2.0.1.0
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getirrthresholdlow`
	
	What it does:
	This command returns the irradiance value (or light level), below which data will not be logged. It is set with `setirrthresholdlow`.
	
	Normal return value(s):
	Minimum irradiance value for data logging, returned in scientific notation. A value of zero (0) can indicate that either the threshold was never set (0 is the default), or it was set to zero.
	
	Example return (for an irradiance value of 73.798):
	`1.25e-4`

`getlogdata` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `getlogdata` What it does: This command returns the log data stored in flash memory. This command can be run during an active logging session (and datalogging will continue) or after a session is stopped with `stoplogdata` or, starting in version 3.0.5.8, after `captureflash` is used with the option to save data to flash. The command will first output three values (total number of values, an integer indicating what information was logged, the logging period, followed by lines of comma-delimited data with date-time-stamp (in seconds since 1970 or "Unix epoch time") followed by all recorded values for that date-time-stamp. Units for Logging Period: Firmware <= 2.0.0.4, with `getlogdata` used after `setlogdata` : Logging period in seconds Firmware >= 2.0.0.5, with `getlogdata` used after `setlogdata` Logging period in 10ms increments, i.e. 1=10ms, 100=1s Firmware >= 3.0.5.8, with `getlogdata` used after `captureflash` Logging period in microseconds, i.e. 10=10us, 1000=1ms Normal return value(s): Total number of Date-Time-Stamp + Value pairs logged Recorded Value Indicator bitmask as follows: 1=Optical Density (x100 for firmware < 2.1.0.0) 2=Percent Transmission (x10 for firmware < 2.1.0.0) 4=Detector Current in amps (picoamps for firmware < 2.1.0.0) 8=Detector Voltage in volts (microvolts for firmware < 2.1.0.0) 16=Device temperature (degrees F) 32=Calibrated Irradiance (see `getirradiance`) Logging Period (see above regarding units) Seconds Since 1970, value #1, value #2, value #n Seconds Since 1970, value #1, value #2, value #n …. Seconds Since 1970, value #1, value #2, value #*n* Error return value(s): -500 if no log data present Notes on returned data values: • If the 100% is not set, both Optical Density and Percent Transmission will return 0. • Log data does not support negative Optical Density values • Calibrated Irradiance will return 0 if there is no calibration data Example return (Notes after the '-' are for documentation purposes and not returned): `5` – total time-stamp + value pairs `4` – detector current `60` – period in seconds (sample every minute) `1378738200, 1.595e-9` – 09 Sep 2013 14:50:00 GMT, 1.595 nanoamps `1378738260, 1.346e-9` – 09 Sep 2013 14:51:00 GMT, 1.346 nanoamps `1378738320, 1.456e-9` – 09 Sep 2013 14:52:00 GMT, 1.456 nanoamps `1378738380, 1.748e-9` – 09 Sep 2013 14:53:00 GMT, 1.748 nanoamps `1378738440, 1.637e-9` – 09 Sep 2013 14:54:00 GMT, 1.637 nanoamps

`getmodelname` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `getmodelname` What it does: This command returns the model name of the device. Normal return value(s): Device model name Error return value(s): N/A

`getod`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getod`
	
	Shortcut:
	`go` (support for this shortcut starting in 3.0.9.4)
	
	What it does:
	This command returns the optical density, multiplied by 100. Because optical density is a relative measurement, this command requires that the 100% (0.00 OD) setting is established with `set100perc`. The maximum optical density returned is based on the device generation as follows:
	
	Gen1: 5.000
	Gen2: 8.000
	Gen3: 11.000
	
	Normal return value(s):
	Optical density (firmware>=2.1.0.0) or Optical Density x 100 (firmware<2.1.0.0)
	
	Error return value(s):
	-500 if the 100% value has not been previously set with `set100perc`
	
	Example return (for an optical density of 1.070):
	`1.070`

`getpeak`
 	
Support Starting in Firmware Version: 3.0.7.0
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getpeak`
	
	What it does:
	This command returns the peak light level that started when `startpeak` was executed.
	
	Normal return value(s):
	Peak light level (units depend on the calibration factor)
	
	Error return value(s):
	-502 if the light level is saturating the detector at the current gain range
	
	Example return (for a peak light level value of 3.465e-4):
	`3.465e-4`

`getpeaks`
 	
Support Starting in Firmware Version: 3.2.2.7
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getpeaks`
	
	What it does:
	This command returns the number of peaks detected between `startintegrate` and `stopintegrate`. A peak is detected when a light level increases 4x over the last sample (i.e. a fast-rising signal).
	
	Normal return value(s):
	The number of peaks detected. Note 0 will be returned if, while the light levels may be peaking, the criteria did not match the 4x factor described above.
	
	Error return value(s):
	None
	
	Example return (for a typical 10 flash alarm beacon test):
	`10`

`getsampletime` Support Starting in Firmware Version: 3.0.5.4 Support Ending in Firmware Version: N/A	Syntax: `getsampletime` What it does: This command returns the sample time in milliseconds (see `setsampletime`). Note that if the sample time is set for automatic (`setsampletime 0`) this command will return the actual sample in time currently in use. Normal return value(s): Sample time in milliseconds, ranging from 10 to 15000 Error return value(s): N/A Example return: `1000` – indicating 1000ms, or 1 second sample time

`getserialnumber`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getserialnumber`
	
	What it does:
	This command returns the serial number of the device. The serial number is stored in one-time-programmable memory at the time of manufacture.
	
	Normal return value(s):
	Device serial number
	
	Error return value(s):
	-500 device serial number has not been set
	
	Example return:
	`10054201208230245`

`gettemp`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`gettemp`
	
	What it does:
	This command returns the internal temperature of the device microcontroller. Note that, while the detector temperature can be loosely inferred from this, this is not equivalent to the detector temperature. The return value is in degrees F, with any units conversion performed by the application software.
	
	Normal return value(s):
	Device microcontroller temperature, in degrees F
	
	Error return value(s):
	N/A
	
	Example return:
	`107`

`gettrans`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`gettrans`
	
	Shortcut:
	`gt` (support for this shortcut starting in 3.0.9.4)
	
	What it does:
	This command returns the percent transmission, multiplied by 10. Because percent transmission is a relative measurement, this command requires that the 100% (0.00 OD) setting is established with `set100perc`.
	
	Normal return value(s):
	Percent Transmission (firmware>=2.1.0.0) or Percent Transmission x 10 (firmware<2.1.0.0)
	
	Error return value(s):
	-500 if the 100% value has not been previously set with `set100perc` or `set100percperm`
	
	Example return (for a percent transmission of 67.3):
	`67.300`

`gettriggerin (Gen3)` Support Starting in Firmware Version: 3.0.5.9 Support Ending in Firmware Version: N/A	Syntax: `gettriggerin [trigger state] [time out seconds]` What it does: This command is for testing of the trigger in line present in some Gen2 and Gen3 devices. The [trigger state] can be "high" (expecting a logic high or "1") or "low" (expecting a logic low or "0"). [time out seconds] is the time to wait for the trigger to appear. Normal return value(s): 0 if trigger not sensed within the time out period 1 if trigger sensed within the time out period Error return value(s): -500 if missing arguments -501 if trigger state is no "low" or "high" -502 if time out period is less than 0 seconds or greater than 300 seconds (5 minutes)

`getuserdark`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getuserdark`
	
	What it does:
	This command returns the dark voltage at the various transimpedance amplifier gain stages, as set by the user with `setuserdark`.
	
	Normal return value(s):
	On Gen1 products: The dark voltage in microvolts, of the two voltage gain stages.
	On Gen2 products: The dark voltage in microvolts, of the three voltage gain stages for each of three feedback resistor stages
	On Gen3 products: The dark voltage in microvolts, of the three voltage gain stages for each of four feedback resistor stages
	
	Error return value(s):
	-500 if the user dark voltage has not been set
	
	Example return:
	Gen1:
	`13014 9832`
	`Gen2:`
	`R1 9735 9607 9564 R2 22885 22746 22670 R3 125018 124804 25190`

`getvagc3`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getvagc3`
	
	What it does:
	This command returns the voltage at the 3rd automatic gain controller (AGC) stage. For "Gen1" products this is a x101 stage. For "Gen2" products, this is a x131 stage.
	
	Normal return value(s):
	AGC3 voltage in volts (firmware >= 2.1.0.0) or microvolts (firmware<2.1.0.0)
	
	Error return value(s):
	N/A
	
	Example return (1.034.. volts):
	`1.034054`

`getvoltage`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getvoltage`
	
	Shortcut:
	`gv` (support for this shortcut starting in 3.0.5.4)
	
	What it does:
	This command returns the voltage output of the transimpedance amplifier, after it is passed through the automatic-gain-control circuit.
	
	Normal return value(s):
	Detector voltage, in volts (firmware >= 2.1.0.0) or microvolts (firmware<2.1.0.0)
	
	Error return value(s):
	N/A
	
	Example return (for 2.415896 volts):
	`2.415896`

`getvoltagestage` Support Starting in Firmware Version: 3.0.5.8 Support Ending in Firmware Version: N/A	Syntax: `getvoltagestage` What it does: This command returns the voltage sensitivity stage in use by the device. Normal return value(s): 1 - 3 Error return value(s): N/A

`getvped`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getvped`
	
	What it does:
	This command returns the "pedestal" voltage for the transimpedance amplifier at the x1 automatic-gain-control stage. This can be used as a diagnostic to verify this important voltage, which should be between roughly 10 and 15 millivolts for Gen1 and Gen2 products.
	
	Normal return value(s):
	"Pedestal" voltage in volts (firmware >= 2.1.0.0) or microvolts (firmware<2.1.0.0)
	
	Error return value(s):
	N/A
	
	Example return (11.087 mV):
	`0.011087`

`getvref`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getvref`
	
	What it does:
	This command returns the reference voltage for the device's A/D converter. This command can be used as a diagnostic if there is ever a suspicion that the voltage reference is faulty. This is should be close to, but not typically exactly, 3.3V.
	
	Normal return value(s):
	Device A/D converter reference voltage, in volts (firmware >= 2.1.0.0) or microvolts (firmware<2.1.0.0)
	
	Error return value(s):
	N/A
	
	Example return:
	`3.291489`

`getvx1`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getvx1`
	
	What it does:
	This command returns the voltage at the x1 automatic-gain-control stage.
	
	Normal return value(s):
	x1 voltage in volts (firmware >= 2.1.0.0) or microvolts (firmware<2.1.0.0)
	
	Error return value(s):
	N/A
	
	Example return (1.543.. volts):
	`1.543087`

`getvx17`
`(Gen2, Gen3)`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getvx17`
	
	What it does:
	This command returns the voltage at the x17 automatic-gain-control stage. It only applies to Gen2 and later devices that have this gain stage
	
	Normal return value(s):
	x17 voltage in volts (firmware >= 2.1.0.0) or microvolts (firmware<2.1.0.0)
	
	Error return value(s):
	-500 if command not supported, i.e. on Gen1 devices
	
	Example return (0.782.. volts):
	`0.782512`

`getwflisten`
`(Gen3)`
 	
Support Starting in Firmware Version: 3.2.2.7
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getwflisten`
	
	What it does:
	This command is used to determine whether or not the system is processing commands coming into the auxiliary serial port (often used for Wi-Fi connectivity).
	
	Normal return value(s):
	0 if the system is not listening on the auxiliary serial port
	1 if the system is listening on the auxiliary serial port
	
	Error return value(s):
	None

`getwifiip`
`(Gen3)`
 	
Support Starting in Firmware Version: 3.2.2.7
 	
Support Ending in Firmware Version: N/A
	`getwifiip`
	
	What it does:
	This command returns the IP address of the device. The command is typically used in conjunction with `setwifi` to ensure an IP address has been assigned at the access point.
	
	Normal return value(s):
	See below.
	
	Error return value(s):
	-500 if Wi-Fi is not supported on the product.
	
	Example return:
	
	CMD
	
	IF=UP
	DHCP=ON
	IP=192.168.0.142:2000
	NM=255.255.255.0
	GW=192.168.0.1
	HOST=0.0.0.0:2000
	PROTO=UDP,
	MTU=1524
	FLAGS=0x40
	TCPMODE=0x0
	BACKUP=0.0.0.0
	<4.41>
	
	EXIT
	
	0
	
	Persist through power-cycle:
	N/A

`getwireless`
`(Gen3)`
 	
Support Starting in Firmware Version: 3.2.2.7
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`getwireless`
	
	What it does:
	This command is used to determine whether or not the onboard Wi-Fi device is enabled. See `setwireless`.
	
	Normal return value(s):
	0 if the Wi-Fi device is disabled
	1 if the Wi-Fi device is enabled
	
	Error return value(s):
	-500 if Wi-Fi is not supported on the product.

`help` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `help` What it does: This command list the most common commands, along with their expected return value. Normal return value(s): List of common commands and return values. Error return value(s): N/A
`set0vbias` Support Starting in Firmware Version: 3.0.5.3 Support Ending in Firmware Version: N/A	Syntax: `set0vbias` What it does: This command sets the bias, on the photodiode anode, to zero volts (unbiased) on devices that support a bias voltage. Normal return value(s): 0 on success Error return value(s): -500 if not supported Persist through power-cycle: No

`set100perc`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`set100perc`
	`set100percperm (saves the value over power cycles)`
	
	What it does:
	Prior to 3.0.5.3, this command read the detectors voltage level (and by inference its current and irradiance level) and sets this as the 100% value to use in Optical Density and %Transmission calculations.
	
	Starting in 3.0.5.3, this command reads the detector current level and sets this as the 100% value to use in Optical Density and %Transmission calculations.
	
	Normal return value(s):
	Prior to 3.0.5.3: The 100% voltage value, in volts (firmware >= 2.1.0.0) or microvolts (firmware<2.1.0.0)
	
	Starting in 3.0.5.3: The 100% current value, in amperes.
	
	Error return value(s):
	
	Prior to 3.0.5.3:
	
	1 if the value is too low to use as a 100% reference voltage, currently set at < 0.020 V
	2 if the value is too high to use as a 100% reference voltage, currently set at > 3.200 V
	
	Starting in 3.0.8.9: Return value of -500 if gain stage saturated
	
	Persist through power-cycle:
	No (`set100perc`)
	Yes (`set100percperm`)
	
	Example return (1.421e-5… current 100% or “full scale”):
	`1.421e-5`

`set5vbias` Support Starting in Firmware Version: 3.0.5.3 Support Ending in Firmware Version: N/A	Syntax: `set5vbias` What it does: This command sets the bias, on the photodiode anode, to -5V volts (negatively biased) on devices that support a bias voltage. Normal return value(s): 0 on success Error return value(s): -500 if not supported Persist through power-cycle: No

`setambientlevel` Support Starting in Firmware Version: 3.0.5.8 Support Ending in Firmware Version: N/A	Syntax: `setambientlevel` What it does: This command sets the existing current and light-level reading as zero, effectively eliminating ambient light/power levels from subsequent readings. For firmware levels<3.2.2.7 This function requires `clearambientlevel` to be issued, followed by a delay to allow the system to accumulate new readings with the ambient level cleared, prior to issuing the `setambientlevel` command. For firmware levels>=3.2.2.7 This function performs a `cearambientlevel`, followed by ten 100ms samples to allow the system to accumulate new readings and exhaust any rolling average, and another ten samples to get a good average for the ambient/zero level. Normal return value(s): 0 on success Error return value(s): -500 if attempting to perform while fast integrating or peak tracking (FW>=3.2.2.7) Persist through power-cycle: No
`setautaveraging` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `setautaveraging` What it does: For firmware levels >= 3.1.4.7 Kept only for backward compatibility and equivalent to setsampletime 0. For firmware levels >= 3.2.2.7 In high-gain “Gen3” devices, the highest gain circuit (“Rf”) utilizes a rolling average that, effectively, extends the sample time in order to improve signal-to-noise ratio. Normal return value(s): 0 on success Error return value(s): N/A Persist through power-cycle: No for firmware < 3.0.5.3 Yes for firmware >= 3.0.5.3

`setcalfactor`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`setcalfactor [calibration factor, 1-20] [description] [sensitivity factor] [saturation current uA]`
	
	What it does:
	This command defines the particular calibration factor details. Details include the calibration factor description and optional units (up to 100 characters), the sensitivity factor (Light Level = Detector Current / Sensitivity Factor), and saturation current in microamps. When defining units, which is used by some software applications, append the units text, after a colon (:), to the description field. See examples below.
	
	For firmware levels < 2.0.0.8:
	`setcalfactor` [calibration factor, 1-20] [desc.] [multiplierx1000] [saturation current uA], where Light Level = Detector Current * multiplier
	
	Normal return value(s):
	0 on success
	
	Error return value(s):
	-500 if missing fields
	-501 if calibration factor is out of the 1-20 range
	-502 if multiplier < 0
	-503 error saving to flash
	
	Persist through power-cycle:
	Yes
	
	Example command (“calfact1” description, 1.3e-7 sensitivity factor, 500 uA saturation current):
	`setcalfactor 1 calfact1 1.3e-7 500`
	
	Example command (“calfact1” description, units of “W/cm2”, 1.3e-7 sensitivity factor, 500 uA saturation current):
	`setcalfactor 1 calfact1:W/cm2 1.3e-7 500`

`setcalfactortemp` Support Starting in Firmware Version: 3.1.3.2 Support Ending in Firmware Version: N/A	Syntax: `setcalfactortemp [calibration factor, 1-20] [description] [sensitivity factor] [saturation current uA]` What it does: This command performs the same function as `setcalfactor`, but does not persist the setting through power cycles. Because the setting does not need to be saved to flash, along with being temporary, it also executes faster.

`setcurrentloop (Gen2, Gen3)`

Support Starting in Firmware Version: 2.0.0.5

`setcurrentloopirr`
`setcurrentloopirr log`
`setcurrentloopdose all`
`setcurrentloopdose alllog`
`setcurrentloopdose sample`
`setcurrentloopdose samplelog`

Support Starting in Firmware Version: 3.2.1.6

Support Ending in Firmware Version: N/A

Syntax:

```
setcurrentloop log
setcurrentloop midpoint
setcurrentloop [min picoamps] [max picoamps]
setcurrentloop [0-24]
setcurrentloopirr [min irradiance] [max irradiance]
setcurrentloopirrlog [min irradiance] [max irradiance]
setcurrentloopdoseall [min dose] [max dose]
setcurrentloopdosealllog [min dose] [max dose]
setcurrentloopdosesample [min dose] [max dose]
setcurrentloopdosesamplelog [min dose] [max dose]
```

IMPORTANT REMINDER: The brackets, [and], are not included in the API call. If included, the value will be interpreted as zero. For example:

`setcurrentloop [12]` will set the current loop to 0 mA.
`setcurrentloop 12` will set the current loop to 12 mA.

What it does:

This command controls the 4-20mA current loop output of devices that support such a current loop.

`setcurrentloop log` sets the device to output a logarithmic scale current that is in relation to the current sensed by the detector. The transfer function is:

Gen2:
4-20mA current = (LOG10(detector current)+8)*3+5
Detector Current = 10^(([4-20mA Current]-5)/3-8)

Gen3:
4-20mA current = (LOG10(detector current)+11)*1+5
Detector Current = 10^(([4-20mA Current]-5)/1-11)

If the detector current is below 10 nA (Gen2) or 1pA (Gen3), the 4-20mA current is set to 4mA. In this mode, the Gen2 is nuanced in that measurements fluctuating around 10nA will result in the 4-20mA output bouncing between 4mA and 5mA (where the default log range starts in the Gen2).

`setcurrentloop midpoint` sets the device to output a linear scale where:

4mA = 0 detector current
12mA = the detector current when this command was set
20mA = double the detector current when this command was set

`setcurrentloop [min picoamps] [max picoamps]` sets the device to output a linear scale where:

4mA = min picoamps
20mA = max picoamps

Note: [min picoamps] must be 25 or greater to avoid defaulting to manual mode (below).

`setcurrentloop [0-24]` sets the device to manually output the current indicated, in milliamps. This is used for testing the current loop.

setcurrentloop (cont)	setcurrentloopirr [min irradiance] [max irradiance] sets the device to output a linear scale where: 4mA = min irradiance 20mA = max irradiance setcurrentloopirrlog [min irradiance] [max irradiance] sets the device to output a log scale where: 4-20mA = (LOG10(irradiance)-LOG10([max irradiance]) * 16 * LOG10([max irradiance]/[min irradiance]) + 20 Irradiance = 10^ ((LOG10([max irradiance]/[min irradiance]) * ([4-20mA Current]-20) / 16 + LOG10([max irradiance])) setcurrentloopdoseall [min dose] [max dose] sets the device to output a linear scale where: 4mA = min dose 20mA = max dose Where dose starts integrating at the earliest of: power on of the device (if setcurrentloopdoseall is set at power on), issuing the startintegrate command, or issuing the setcurrentloopdoseall command setcurrentloopdosealllog [min dose] [max dose] sets the device to output a log scale where: 4-20mA = (LOG10(dose)-LOG10([max dose]) * 16 * LOG10([max dose]/[min dose]) + 20 Dose = 10^ ((LOG10([max dose]/[min dose]) * ([4-20mA Current]-20) / 16 + LOG10([max dose])) Where dose starts integrating at the earliest of: power on of the device (if setcurrentloopdoseall is set at power on), issuing the startintegrate command, or issuing the setcurrentloopdoseall command

setcurrentloop (cont)	setcurrentloopdosesample [min dose] [max dose] sets the device to output a linear scale where: 4mA = min dose 20mA = max dose Where dose is calculated over the time period set by setsampletime or setsampletimetemp setcurrentloopdosesamplelog [min dose] [max dose] sets the device to output a log scale where: 4-20mA = (LOG10(dose)-LOG10([max dose]) * 16 * LOG10([max dose]/[min dose]) + 20 Dose = 10^ ((LOG10([max dose]/[min dose]) * ([4-20mA Current]-20) / 16 + LOG10([max dose])) Where dose is calculated over the time period set by setsampletime or setsampletimetemp Special Current Loop Values: 3.25mA Loop issue 1.00mA Calibration factor not selected for setcurrentloopirr/dose 2.00mA Detector is saturated Normal return value(s): 0 on success Error return value(s): -500 if command not supported, i.e. on Gen1 devices -501 if missing fields -502 if there is a bad current loop value (applies to setcurrentloop [0-24]) Persist through power-cycle: Yes for all but setcurrentloop [0-24], which is intended as more of a test function. Example command (4-20mA = 100 to 700 calibrated units of irradiance): setcurrentloopirr 100 700 Example command (set current loop to 12mA for testing): setcurrentloop 12

`setdatetime`
`(Gen2, Gen3)`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`setdatetime 12/05/2013 19:02:05`
	
	What it does:
	This command sets the real time clock's date and time. It accepts either of the following formats:
	mm/dd/yyyy hh:mm:ss
	mm/dd/yy hh:mm:ss
	
	The device is designed such that this date/time setting is UTC/GMT time. The device stores all date/timestamps in Epoch time format, which is later read out and converted to local time by an application. This is the case with the Data Logger software.
	
	Normal return value(s):
	0 on success
	
	Error return value(s):
	
	-500 if command not supported
	-501 if missing fields
	-502 if parameters are out of range
	
	Persist through power-cycle:
	Yes, assuming coin/cell battery is in place with adequate charge

`setfriendlyname` Support Starting in Firmware Version: 3.0.5.4 Support Ending in Firmware Version: N/A	Syntax: `setfriendlyname [friendly name, up to 30 characters]` What it does: This command sets the "friendly name" of the device. The friendly name is used by applications to help users easily identify which device is in use. Normal return value(s): 0 on success Error return value(s): -500 if missing fields -501 if error storing value Persist through power-cycle: Yes

`sethiaveraging` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `sethiaveraging` What it does: For firmware levels >= 3.1.4.7 Kept only for backward compatibility and equivalent to setsampletime 2000. For firmware levels >= 3.2.2.7 In high-gain "Gen3" devices, the highest gain circuit ("Rf") utilizes a rolling average that, effectively, extends the sample time in order to improve signal-to-noise ratio. Normal return value(s): 0 on success Error return value(s): N/A Persist through power-cycle: No for firmware < 3.0.5.3 Yes for firmware >= 3.0.5.3
`setirrthresholdlow` Support Starting in Firmware Version: 2.0.1.0 Support Ending in Firmware Version: N/A	Syntax: `setirrthresholdlow [calibrated reading]` What it does: This command sets the minimum irradiance value (calibrated reading) to use in conjunction with data logging. When set, and irradiance is being monitored with data logging, data will not be recorded unless the irradiance level meets this threshold. Normal return value(s): 0 on success Error return value(s): -500 if missing fields -501 if parameters are out of range: Less than 0 or greater than 1 for firmware 2.0.1.0 -> 3.0.5.8 Less than 0 or greater than 1000 for firmware >= 3.0.5.9 -502 if error saving to flash memory Persist through power-cycle: Yes

`setlowaveraging` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `setlowaveraging` What it does: For firmware levels >= 3.1.4.7 Kept only for backward compatibility and equivalent to setsampletime 200. For firmware levels >= 3.2.2.7 In high-gain "Gen3" devices, the highest gain circuit ("Rf") utilizes a rolling average that, effectively, extends the sample time in order to improve signal-to-noise ratio. Normal return value(s): 0 on success Error return value(s): N/A Persist through power-cycle: No for firmware < 3.0.5.3 Yes for firmware >= 3.0.5.3
`setmedaveraging` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `setmedaveraging` What it does: For firmware levels >= 3.1.4.7 Kept only for backward compatibility and equivalent to setsampletime 500. For firmware levels >= 3.2.2.7 In high-gain "Gen3" devices, the highest gain circuit ("Rf") utilizes a rolling average that, effectively, extends the sample time in order to improve signal-to-noise ratio. Normal return value(s): 0 on success Error return value(s): N/A Persist through power-cycle: No for firmware < 3.0.5.3 Yes for firmware >= 3.0.5.3

`setsampletime`
 	
Support Starting in Firmware Version: 3.0.5.4
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`setsampletime 0`
	
	`setsampletime` [10 - 15000]
	
	What it does:
	This command sets the sample time used by the Analog-to-Digital converter when reading the voltage from the transimpedance amplifier in milliseconds. `setsampletime 0` sets the sample time to automatic, resulting in 500ms for high level signals and up to 2 seconds for lower signals.
	
	For typical monitoring applications, sample time is rarely set below 250ms and typically at 500ms or 1000ms (1s). For high-speed sampling in conjunction with data logging (see `startdatalog`), sample times as low as 10ms can be used.
	
	Normal return value(s):
	0 on success
	
	Error return value(s):
	-500 if missing fields
	-501 if value is out of range
	
	Persist through power-cycle:
	Yes
	
	Example command (for auto sample time):
	`setsampletime 0`
	
	Example command (for 50ms sample time):
	`setsamplecount 50`

`setsampletimetemp` Support Starting in Firmware Version: 3.0.6.7 Support Ending in Firmware Version: N/A	Syntax: `setsampletimetemp 0` `setsampletimetemp` [10 - 15000] What it does: This command performs the same function as `setsampletime`, but does not persist the setting through power cycles. Because the setting does not need to be saved to flash, along with being temporary, it also executes faster.

`settriggerout (Gen3)`
 	
Support Starting in Firmware Version: 3.0.5.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`settriggerout on`
	`settriggerout off`
	
	What it does:
	This command sets the output trigger line to either a logic 1 (“on”) or a logic 0 (“off”). This command is used for testing purposes, with the trigger out circuit being exercised as part of the captureflash command.
	
	Normal return value(s):
	0 on success
	
	Error return value(s):
	-500 if missing fields
	-501 if bad arguments
	
	Persist through power-cycle:
	No

`setuserdark`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`setuserdark`
	
	What it does:
	This command captures the detector voltage signal, at all automatic-gain-control stages, and stores the values in device flash memory. When used in conjunction with `useuserdark`, all subsequent readings will have this value removed automatically. Starting in firmware version 2.0.0.5, if the mode is already set to `useuserdark` then the new values will automatically be applied.
	
	Normal return value(s):
	User dark value, in microvolts, for all gain stages (separated by a space)
	
	Error return value(s):
	-500 if error saving setting to flash memory
	
	Persist through power-cycle:
	Yes
	
	Example return:
	Gen1:
	`13014 9832`
	Gen2:
	`R1 9735 9607 9564 R2 22885 22746 22670 R3 125018 124804 25190`

`setwifi`
`(Gen3)`
 	
Support Starting in Firmware Version: 3.2.2.7
 	
Support Ending in Firmware Version: N/A
	`setwifi [ssid] [password]`
	
	What it does:
	This command programs the onboard Wi-Fi chip set with the SSID and password of the access point. This can be used on non-Windows system where the NetConfig application is not supported, and is often run from within a terminal emulator.
	
	Normal return value(s):
	See below.
	
	Error return value(s):
	-500 if missing arguments
	
	Example usage:
	`setwifi Bobs-iPhone8 pword9273`
	
	Example return (Note that, initially, the interface is still down as indicated by "IF=DOWN". See `getwifiip` to verify interface is up after this command is complete) :
	
	`CMD`
	
	`AOK`
	`<4.41>`
	
	`AOK`
	`<4.41>`
	
	`Storing in config`
	`<4.41>`
	
	`*Reboot*`
	`CMD`
	
	`IF=DOWN`
	`DHCP=ON`
	`IP=0.0.0.0:2000`
	`NM=0.0.0.0`
	`GW=0.0.0.0`
	`HOST=0.0.0.0:2000`
	`PROTO=UDP,`
	`MTU=1524`
	`FLAGS=0x40`
	`TCPMODE=0x0`
	`BACKUP=0.0.0.0`
	`<4.41>`
	
	`EXIT`
	
	`0`
	
	Persist through power-cycle:
	Yes

`setwireless`
 	
`(Gen3)`
 	
Support Starting in Firmware Version: 3.0.6.1
 	
Support Ending in Firmware Version: N/A
	`setwireless on`
	`setwireless off`
	
	What it does:
	This command enables or disables the wireless functionality. When disabling the functionality, the system will immediately shut down the wireless part. When enabling the wireless functionality the system requires a power cycle to restore the wireless operation.
	
	Normal return value(s):
	0 on success
	
	Error return value(s):
	-500 if missing arguments
	-501 if not supported based on the device generation
	-502 if bad syntax/arguments
	
	Persist through power-cycle:
	Yes

`startintegrate`
 	
Support Starting in Firmware Version: 3.0.7.0
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`startintegrate`
	`startintegrate [max Rf number when using Auto Rf]`
	
	What it does:
	This command starts integrating the light level at high-speed to capture fast pulses. The integrated light-level is returned with `getintegrate` and the integration is halted with `stopintegrate`. Note that `startintegrate` always resets the integrated value to 0.
	
	Also see `getpeaks`.
	
	Starting in firmware version 3.2.2.7, the command takes an optional parameter defining the maximum feedback resistor (Rf) number to be used if Auto Rf is enabled. Furthermore, during the integration cycle between `startintegrate` and `stopintegrate`, the Rf value will only be lowered to repeatedly detect the brightest light-level. This is used for multiple flash tests where the light level may vary over many decades, such as emergency beacon testing. For example `startintegrate 2` will result in the system (a) not ranging below Rf=2 and (b) never increasing the Rf number after the flash level drops in anticipation of capturing the next flash.
	
	Normal return value(s):
	0 on success
	
	Error return value(s):
	None

`startlogdata` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `startlogdata` [variable bitmask] [logging period – see below] [seconds since 1970] What it does: This command defines the logging parameters, and immediately starts logging. This command cannot be run when either a logging session is already active, a session has been stopped (with `stoplogdata`) but the log data has not yet been erased with `eraselogdata`,`or` a captureflash session has completed with saved data that has not yet been erased with `eraselogdata`. The command takes three parameters, with a space between parameters, as follows: [variable bit mask] 1=Optical Density (x100) 2=Percent Transmission (x10) 4=Detector Current (picoamps) 8=Detector Voltage (microvolts) 16=Device temperature (degrees F) 32=Calibrated Irradiance (see `getirradiance, setirrthresholdlow`) 128=Use Real Time Timestamps (as opposed to relative time stamps, Gen2 only) *For firmware versions 2.0.0.1 and earlier:* [logging period] = desired logging period, in seconds, divided by 10 A 1, for example, would indicate a 10 second delay between logging; a value of 360 would indicate a 3600 second delay, or 1 hour, between log entries. The maximum value is 8640, which is equivalent to 86400 seconds or 1 day. Any values above this will results in a 1 day logging period. *For firmware versions 2.0.0.2 and later* [logging period] = desired logging period in seconds A 1, for example, would indicate a 1 second delay between logging; a value of 3600 would indicate a 3600 second delay, or 1 hour, between log entries. The maximum value is 86400, which is equivalent to 86400 seconds or 1 day. Any values above this will results in a 1 day logging period. *For firmware versions 2.0.1.0 and later* [logging period] = desired logging period, in 10 milliseconds increments. A 1, for example, would indicate a 10 millisecond delay between logging, or 100Hz logging. A value of 6000 would indicate a 1 minute delay. The maximum value is 8640000, which is equivalent to 86400 seconds or 1 day. Any values above this will results in a 1 day logging period. A value of 0 will default to a 10ms delay. NOTE: there is a known bug that limits the maximum log period to 14.4 minutes. See your product representative for information on an update to resolve this issue if needed. [seconds since 1970] This is “Unix epoch time”, with most application coding environments having a mechanism to convert a date-time structure to epoch time. This is ignored, and should be set to 0, when using real-time timestamps. Normal return value(s): 0 on success

`startlogdata (cont)`	Error return value(s):
	-500 missing parameters
	-501 session already started. Must use `stoplogdata` and `eraselogdata` to start new.
	-502 errors with the variable bit mask
	
	Persist through power-cycle:
	Yes.
	
	Common usage is to use startlogdata, disconnect the device from the computer, connect the device to a battery or AC-power in the lab or field, and the device will continue logging on power up using time-stamps that are relative to the start time. In this scenario care must be taken to ensure any power down time is negligible to the desired time-stamp accuracy. Gen2 devices have the option of using a battery-back real-time clock (see 128 bitmask value above).
	
	Example command (to log detector current and device temperature, every 10ms (Firmware 2.0.1.0), starting at 09 Sep 2013 14:50:00 GMT. Note bitmask of 20 is 4/current + 16/temperature):
	`startlogdata 20 1 1378738200`
	
	Example command. Same as above, but sampling every minute (60 seconds is 6000 10ms increments) and adding 128 to the bit mask for use of a real time clock on a Gen2 device):
	`startlogdata 148 6000 0`

`startpeak`
 	
Support Starting in Firmware Version: 3.0.7.0
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`startpeak`
	
	Version 3.1.2.3 and later provide optional syntax:
	`startpeak [rising edge multiplier] [capture time in ms]`
	
	Version 3.1.3.3 and later provide optional syntax:
	`startpeak [falling edge multiplier] [capture in ms]`
	
	Version 3.2.2.7 and later provide optional syntax:
	`startpeak [capture in ms]`
	
	What it does:
	`startpeak`
	
	This command starts peak detection of the light level at high-speed to capture fast pulses of light. The peak light level is returned with `getpeak` and the peak detection is halted with `stoppeak`. Note that `startpeak` always resets the peak value to 0. Peak detection can detect very fast peaks, in the sub-microsecond range, due to the RC peak-delay circuit on the front-end of the device. Best performance is achieved when the feedback resistor range is fixed, i.e. usefeedbackres 1, usefeedbackres 2, etc.
	
	`startpeak [capture time in ms]`
	
	The new syntax released in Firmware Version 3.2.2.7 allows the API to (a) wait on a rising edge (with a default peak rising multiplier of 5) and return the integral calculated for the duration defined by [capture in ms].
	
	`startpeak [rising edge multiplier] [capture time in ms]`
	
	The new syntax released in Firmware Version 3.1.2.3 allows the API to (a) wait on a rising edge (defined by the [peak rising multiplier] variable), and (b) return data time, value comma-separated data to characterize the entire curve for a duration defined by [capture time in ms]. The normal usage of this syntax is to issue startpeak along with its command line arguments, for example “startpeak 30 100”, and wait for the data to return, which will happen automatically after rising edge detection. This can be used to capture a rise in light level as well as a peak and decay. Timeouts awaiting rising detection need to be handled by the application code, and issue a “stoppeak” after a timeout period that is agreeable for the particular application. The time, value data pairs are designed to return higher-resolution data to characterize the rise and, for peak detection, immediate fall, followed by lower-resolution data to characterize the decay. As such, the first 60 data points are captured as quickly as possible, approximately 80uS apart, followed by the remaining points captured at 5ms intervals until the [peak decay timer in ms] parameter is reached.
	
	`startpeak [falling edge multiplier] [capture time in ms]`
	
	This syntax was added in 3.1.3.1 to trigger the capture based on the falling edge (lower light or darkening). [falling edge multiplier] must be between 0.001 and 1 and indicates the multiplier to be applied to the running peak in order to trigger the capture.

`startpeak (cont.)`	Normal return value(s):
	`startpeak`
	0 on success
	
	`startpeak [capture in ms]`
	0 on successful start of the command, followed by:
	[Integral],[Peak] values of flash after flash is acquired and integrated
	
	`startpeak [rising edge multiplier] [capture time in ms]`
	`startpeak [falling edge multiplier] [capture time in ms]`
	0 on successful start of the command, followed by time, value pairs related to the shape of the curve after rising or falling edge detection. The first 60 data points are returned at approximately 80uS intervals, while the remaining points are returned at 5ms intervals.
	
	The literal END will be sent after the last time, value pair to indicate to the polling routine that the data is complete.
	
	Error return value(s):
	
	`startpeak [peak detect multiplier] [peak decay timer in ms]`
	-500 will be returned as the last data point in the peak data if a saturation was detected during the peak detection

`stopintegrate` Support Starting in Firmware Version: 3.0.7.0 Support Ending in Firmware Version: N/A	Syntax: `stopintegrate` What it does: This command stops the integration started with `startintegrate`. See also `getintegrate`. Normal return value(s): 0 on success Error return value(s): -500 if light level integration has not been started with `startintegrate` -501 if setcurrentloopdose… API has been set (integrate must remain on)
`stoplogdata` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `stoplogdata` What it does: This command stops a log session that was started with `startlogdata`. Note that, because date-time-stamping is relative to the start time set with `startlogdata`, logging cannot be restarted after a `stoplogdata`. Instead, log data must first be erased using `eraselogdata`. Normal return value(s): 0 on success Error return value(s): -500 if no active logging session to stop

`stoppeak` Support Starting in Firmware Version: 3.0.7.0 Support Ending in Firmware Version: N/A	Syntax: `stoppeak` What it does: This command stops the peak detection started with `startpeak`. See also `getpeak`. Normal return value(s): 0 on success Error return value(s): -500 if peak tracking has not been started with `startpeak`
`stream` Support Starting in Firmware Version: 3.1.2.3 Support Ending in Firmware Version: N/A	Syntax: `stream [type] [number of samples]` What it does: This command is the fastest way to actively and continuously capture data from a device. While captureflash is the fastest mechanism to capture 4096 data points on the device, the stream API is the fastest way to continuously stream data to the device. The data rate is approximately 500 data points per second. Note that the function returns values only, as opposed to time, value pairs. As a result it is up to the calling application to timestamp the data. In Firmware Version 3.1.2.3, the steam function would not change feedback resistors when auto-gain range was selected. Starting in Firmware Version 3.1.2.4, the auto-gain function will be executed when `usefeedbackres` is set to zero. [type] 0= Detector Voltage (volts) 1= Detector Current (amps) 2=Light Level (custom units) [number of samples] 1 – 10000, the number of samples streamed in return Normal return value(s): A series of values in scientific notation, one per line. Error return value(s): -500 if missing fields -501 if bad stream type (>2) or number of samples (>10000) -502 if factor not defined (cannot monitor light level without a factor) Persist through power-cycle: N/A

`usecalfactor` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `usecalfactor` [calibration number, 0-20] What it does: This command selects a particular calibration factor (see `setcalfactor`), which in turn will define the detector current-to-irradiance multiplier as well as the detector saturation current. A calibration number of 0 is a special case. Using 0 results in no longer using any calibration factors. When this is done, `getirradiance` will only return a value if `setsimpleirrcal` or `setirrdatapoint` has been used. Normal return value(s): 0 on success Error return value(s): -500 if missing fields -501 if factor number is outside the 0-20 range -502 if factor not defined -503 if error saving to flash Persist through power-cycle: Yes (firmware Version 2.0.0.0 and later), No otherwise Example command: `usecalfactor 5`
`usecalfactortemp` Support Starting in Firmware Version: 3.0.6.7 Support Ending in Firmware Version: N/A	Syntax: `usecalfactortemp` [calibration number, 0-20] What it does: This command performs the same function as `usecalfactor`, but does not persist the setting through power cycles. Because the setting does not need to be saved to flash, along with being temporary, it also executes faster.
`usefactorydark` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `usefactorydark` What it does: This command will cause all subsequent readings to remove the factory dark value before presenting any voltage, current, etc. readings. This is the default dark setting upon power up. Normal return value(s): 0 on success Error return value(s): -500 if there has been no factory dark value set Persist through power-cycle: No

`usefeedbackres (Gen2, Gen3)`
 	
Support Starting in Firmware Version: 2.0.0.3
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`usefeedbackres [resistor selection 0-4]`
	`usefeedbackres`
	
	What it does:
	For Gen2 devices, which have 3 available feedback resistors, this command selects the resistor to be used as follows:
	
	0: have the device automatically select a resistor based on light-level
	1: use feedback resistor #1, usually the lowest value resistor
	2: use feedback resistor #2
	3: use feedback resistor #3
	4: use feedbac resistor #4 (Gen3 only)
	
	Starting in 3.0.5.4, issuing usefeedbackres without parameters returns the value of the feedback resistor in use [0=feedback resistor 1, 1=feedback resistor 2, etc.]
	
	Starting in 3.0.6.7, issuing usefeedbackres without parameters returns the value of usefeedbackres setting [0=auto, 1=fixed to feedback resistor 1, 2=fixed to feedback resistor 2, etc.].
	
	Normal return value(s):
	0 on success (for the first syntax above)
	0-4, depending on the feedback resistor directive (for the second syntax above)
	
	Error return value(s):
	-500 if missing parameters (pre 3.0.5.4)
	-501 if the device does not support multiple feedback resistors
	-502 it the resistor selection value out of range
	-503 resistor selected, but error saving the change to flash memory
	
	Persist through power-cycle:
	Yes

`usefeedbackrestemp`
 	
Support Starting in Firmware Version: 3.0.6.7
 	
Support Ending in Firmware Version: N/A
	Syntax:
	`usefeedbackrestemp [resistor selection 0-4]`
	`usefeedbackrestemp`
	
	What it does:
	This command performs the same function as `usefeedbackres`, but does not persist the setting through power cycles. Because the setting does not need to be saved to flash, along with being temporary, it also executes faster.

`usenodark` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `usenodark` What it does: This command will eliminate any dark current consideration. Normal return value(s): 0 on success Error return value(s): N/A Persist through power-cycle: No
`useuserdark` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: N/A	Syntax: `useuserdark` What it does: This command will cause all subsequent readings to remove the user dark value (see `setuserdark`) before presenting any voltage, current, etc. readings. Normal return value(s): 0 on success Error return value(s): -500 if there has been no user dark value set Persist through power-cycle: No

`echooff` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version:3.2.2.7	Syntax: `echooff` What it does: This command enters a mode where only values are sent back in response to a command. This is the default mode on power up of the device, and the mode used for programming by an application. Normal return value: 0 on success Error return value(s): N/A Persist through power-cycle: No
`echoon` Support Starting in Firmware Version: 2.0.0.3 Support Ending in Firmware Version: 3.2.2.7	Syntax: `echoon` What is does: This command enters a verbose mode whereby contextual help is echoed back for each command completion. The mode is useful when interacting with the device from a terminal server or from the CLI program. Normal return value: 0 on success Error return value(s): N/A Persist through power-cycle: No

`setsamplecount`
 	
Support Starting in Firmware Version: 2.0.0.6
 	
Support Ending in Firmware Version: 3.0.5.3
	Syntax:
	`setsamplecount` [1 – 200]
	
	What it does:
	This command sets the number of samples taken by the Analog-to-Digital converter when reading the voltage from the transimpedance amplifier. This is the immiedate averaging done on the input signal, before additional averaging is done by the `set*averaging` commands. Typically a value of 200 is used (the default) for standard operation, or a value of 1 is used for high-speed sampling (up to 100 samples/sec). WARNING: modifying this value will impact the communication between the meter and any device talking to it over the USB port. Talk to the manufacturer for details.
	
	Normal return value(s):
	0 on success
	
	Error return value(s):
	-500 if missing fields
	-501 if value is out of range
	
	Persist through power-cycle:
	Yes
	
	Example command (for sampling at 100/sec):
	`setsamplecount 1`
	
	Example command (for default sampling rate):
	`setsamplecount 200`