Reference Guide

Ayla Module CLI Specifications

Version: 9.0

Date Released: September 19, 2017

Document Number: AY006UCL3-9

Ayla Networks

Referenced Documents

Ayla Networks does not supply all documents that are referenced in this document with the equipment. Ayla Networks reserves the right to decide which documents are supplied with products and services.

Contact Information

Ayla Networks TECHNICAL SUPPORT and SALES

Contact Technical Support: https://support.aylanetworks.com
or via email at support@aylanetworks.com

Contact Sales: https://www.aylanetworks.com/company/contact-us

Ayla Networks REGIONAL OFFICES

GREATER CHINA
Shenzhen
Room 310-311
City University of Hong Kong
Research Institute Building
No. 8 Yuexing 1st Road
High-Tech Industrial Park
Nanshan District
Shenzhen, China
Phone: 0755-86581520

HEADQUARTERS
Silicon Valley
4250 Burton Drive, Suite 100
Santa Clara, CA 95054
United States
Phone: +1 408 830 9844
Fax: +1 408 716 2621

JAPAN
Wise Next Shin
Yokohama, 2-5-14
Shnyokohama, Kohokuku
Yokohama-shi, Kanagawa-ken
Yokohoma, 222-0033 Japan

EUROPE
Munich
Ludwigstr. 8
D-80539 München,
Germany

TAIWAN
Taipei
5F No. 250 Sec. 1
Neihu Road, Neihu District
Taipei 11493, Taiwan

For a Complete Contact List of Our Offices in the US, China, Europe, Taiwan, and Japan:
https://www.aylanetworks.com/company/contact-us

Table of Contents

1 Introduction

This document provides information on the command line interface (CLI) used on the Ayla Module's console serial port.

The console serial port is usually connected to an FTDI serial-to-USB port. It runs at 115,200 bits/sec, 8-bits, 1 stop bit, no parity.

1.1 Audience

This document is for developers who want to familiarize themselves with Ayla Module CLI.

1.2 Related Documentation

- Module and MCU Interface Specification (AYOO6MSP3)
- Design Kit - USB and TTL Serial Communication User Manual (AY006TTL2)

2 Commands

The following is a list of the commands available for use with the module. The CLI is accessed through the serial port.

The CLI is mostly used by ayla_setup script during module manufacturing initialization, testing, and to set the factory configuration.

This is a listing of CLI commands:

- client - set client attributes
- eth - set ethernet driver attributes
- file- set contents of a file (e.g., certificate, key)
- gpio - setup GPIO mode attributes
- host - setup the interface to MCU
- hw - hardware platform specific commands
- id - set ID fields (mfg mode only)
- img - image management (4390 only)
- locale - change locale string attributes (OBSOLETE)
- log - adjust logging attributes
- log-client - enable/disable log-client
- log-snap – check crash logs saved by the module
- mfg_mode - disable mfg mode (mfg mode only)
- mfi – set MFI configuration attributes
- oem - set oem strings
- power - control power management attributes
- reset - reset the module
- rssi - show RSSI and antenna attributes
- run - run another program (e.g., flup)
- save - save the configuration
- sched - configure the names of the schedules
- secure - enable boot security
- server - enable/disable AP mode property test and security

- setup_mode - disable setup mode.
- show - show part of configuraton (e.g., id)
- time - see UTC and module time attributes
- version - display module image and booter version
- wifi - set Wi-Fi attributes

client commands

`client <enable|disable>`

Controls whether the client is enabled or disabled.

`client poll_interval <# of secs>`

Sets the frequency of the client polling ADS for updates and checks for internet connectivity if the notification service is no longer reachable. Default = 300 seconds.

If set to 0, no polling takes place (useful for modules without to-device properties).

`client`

Displays the current client configuration.

`client test`

Sets a mode that indicates to ADS that the connections are being done for OEM testing and not counted as the first connection from the device by an end-user.

This setting is not persisted and lasts only until the next reboot.

`client server region <region code>`

Sets the region for the device, which determines the set of ADS servers that the device should use.

This command requires that the device be in setup mode, and a save and a reset be performed after the command is issued. The device's DSN and other information must be present on the service for that region for this command to succeed. Current supported region values include "CN" for China and "US" for everywhere else. If set to the empty string, the region will be set to the default (US).

eth command

`eth [enable|disable]`

Enable or disable Ethernet capability on Ayla module.

`eth mac_addr <mac address> i.e. eth mac_addr 5e:c3:32:32:ac`

Set MAC address to be used for Ethernet protocol by Ayla Module.

`ethi spi_speed [0..8]`

The spi_speed values are platform-dependent and should be recommended based on hardware restrictions. The value 0 selects the compiled-in default.

On STM32F2xx-based modules, the spi_speed values 1 thru 8 translate to a SPI clock frequency which is the bus frequency divided by 2^n. For example, 1 means divide by 2, 2 means divide by 4, ... up to 8 meaning divide by 128. The fastest speed 1, is 15 MHz. Default for STM32F2xx is 7.5 MHz.

file commands

`file start <file-number>`

Clears file 0 and makes it the current file for the "file add" command.

`file add <data>`

Adds data, which may be quoted, to the file.

`file crc <file-number>`

"file crc 0" will show the length and CRC-32 of file 0, to allow for verification of correct transfer.

gpio commands

GPIO mode allows a module to associate properties with I/O pins on the module. This can be useful in simple applications that don't require a separate MCU to perform the product functions. A module in GPIO mode does not support the SPI interface to an MCU.

With no arguments, gpio displays the configured and dynamic state of all configurable pins.

With a single pin or property name argument, the command shows the configured and dynamic state of that pin.

GPIO Options

`analog`

Sets the pin as analog. Digital is the default.

`in`

Sets the pin as an input (default).

`inv`

Sets the pin as *active_low*. The logic is inverted; a low-voltage produces a 1 for the property and vice-versa.

`od`

Sets the pin as an *open-drain* output. If not specified, the output is *push-pull*.

`out`

Sets the pin as an output.

`persist`

For an output pin, this specifies the property value is saved so the most recent output pin setting is restored if the device loses power or is restarted.

To avoid saving rapid-fire changes, the setting may only persist after a short delay.

`pd`

Assigns the pin a *pull-down* resistor. This is most useful for input pins.

`prop <name>`

(optional) Sets the property pin name.

`pu`

Assigns a *pull-up* register to the pin. This is usually used for input pins. However, some devices may be capable of supplying them on *open-drain* pins.

`status`

The selected output pin should be driven as a module status LED.

The keyword "wifi" selects the Wi-Fi or network status, and the keyword "server" selects the ADS service connection status. These work as defined in the Smart Plug marketing requirements.

`toggle <pin>`

When used with an input pin, indicates the pin will toggle the specified output pin.

If the input pin is *active_low*, the output pin will be toggled after the input pin goes low.

GPIO Syntax and Examples

The CLI command allows the module in setup mode to be configured for the device. The command may also be used in any mode to display the current I/O state. When enabling GPIO for the first time, it is necessary to reset after saving so that the SPI bus is not initialized.

```
gpio [enable|disable]
gpio <pin> in [prop <name>] [inv] [toggle|set|r<out-pin>] [pu|pd]
gpio <pin> [prop <name>] [analog] [in|out] [inv] [od] [pu|pd] [toggle
<pin>] [val <value>] [persist]
gpio <pin> out [prop <name>] [od] [inv] [persist] [def <value>]
gpio status [wifi | service] <pin>
gpio [ready | link | intr | fact_rst | reg | mcu_rst] <pin>
gpio wps <pin>
gpio wifi_setup_opt <pin>
```

```
gpio commit
gpio
gpio <pin>
gpio <property>
```

GPIO mode can only be changed in setup or manufacturing mode. The GPIO settings can be displayed in user mode for diagnostic purposes. GPIO settings may not be completely effective until the 'gpio commit' command has been entered or the configuration has been saved and the module reset.

`gpio enable`

Turns on GPIO mode.

`gpio disable`

Turns off GPIO mode.

GPIO mode will not be turned off unless the user performs a 'save' and a 'reset'.

`gpio`

To show available pins, after enabling GPIO mode, type 'gpio' without arguments. To show the setting for a single pin, give its pin name or the associated property name as the only argument.

`gpio <pin> in [prop <name>] [inv] [toggle <out-pin>] [pu|pd]`

`gpio <pin> out [prop <name>] [od] [inv]`

Sets up a GPIO as an input or output pin, with arguments as follows:

`in`

Sets up an input pin

`out`

Sets up an output pin

`prop <name>`

Associates the boolean property called <name> with the input or output pin.

If an input pin, the property will be set to 1 when the signal is active and 0 when inactive. Signal transitions will be debounced (i.e. filtered) so that only changes that occur at least 15 ms after previous changes will be sent to the service.

If an output pin, the pin's value will be set based on the property's value. When the property is received from the service or a LAN client and the value received is 1, the associated output pin is activated. It is deactivated if the value received is 0.

`inv`

The presence of this keyword indicates the signal on the pin is *active-low*.

`toggle`

When set on an input pin, the toggle keyword allows an output pin to be toggled whenever the input transitions from inactive to active.

`pu`

Selects a *pull-up* resistor.

`pd`

Selects a *pull-down* resistor.

`od`

Selects *open-drain* mode on an output pin. Otherwise the mode is push-pull.

`gpio [status] wifi <pin>`

or

`gpio [status] service <pin>`

Sets up status outputs. Use one of these commands after the output pin is configured.

These would normally control orange and green LEDs, respectively, perhaps in a bi-color package. They function as specified in the Smartplug requirements document.

`gpio ready <pin>`

Configures the READY_N pin to something other than the default defined in the module's pin mapping specification.

The READY_N signal from the module is asserted to indicate that the module has initialized the SPI interface and is ready to accept commands from the attached MCU. The signal is only used when the host mode of the module is set to SPI.

`gpio link <pin>`

Configures the LINK_N pin to something other than the default defined in the module's pin mapping specification.

The LINK_N signal from the module is asserted to indicate that the module has connected to the Ayla Device Service.

`gpio intr <pin>`

Configures the INTR_N pin to something other than the default defined in the module's pin mapping specification.

The INTR_N signal from the module is asserted when the module has a SPI message waiting to be received by the MCU. The signal de-asserts after the message has started to transfer. Each time a SPI message becomes pending, the module generates a falling edge on INTR_N, even if it is already low.

`gpio fact_rst <pin>`

Sets up a factory configuration reset input. Use this after configuring the pin.

This input is checked shortly after power on. If it is set for 5 contiguous seconds after 'power on', the device will perform a factory configuration reset. During the 5 second period, the status LEDs blink alternately as long as the input is asserted.

`gpio reg <pin>`

Sets up an input that starts a device registration window with the Ayla Device Service. Use this after configuring the pin.

This button will be effective only when the device is indicating that it is not registered, and currently only if there is a service status output configured.

`gpio mcu_rst <pin>`

This pin may be used to remotely reset a Host MCU using the dashboard. The hard reset on the dashboard will toggle this pin for 100ms. OEMs may use this to reset their MCU by configuring the pin as output with *open-drain* and then using an external resistor to pull-up the line. The mcu_rst feature is only present in **bc-2.3** onwards.

`gpio wifi_setup_opt <pin>`

Configures a push button for the alternative wifi setup mode (other than AP mode).

Currently, Ayla only supports the WeChat's Airkiss protocol.

`gpio wps <pin>`

Configures a push button for starting WPS.

The `wifi wps` command must also be issued to enable Wi-Fi configuration using this method. AP's WPS button has to be pressed as well, and then AP will allow module to join it without entering password manually. Time window for this configuration to complete is less than 2 minutes.

NOTE Also see the Appendix A in this document for additional information about GPIO I/O and GPIO Configuration for SmartPlug

GPIO Test Mode commands

GPIO Test Mode enables the features of a device using GPIO mode to be tested during manufacturing. Testing is possible after assembly, without access to the serial console, without a connection to ADS, without connecting to a Wi-Fi Access Point and without the use of a mobile app. Testing is initiated using a designated push-button on the device.

A complete description of GPIO Test Mode behavior is described in the GPIO Test Mode Application Note.

`gpio test <pin>`

Designates the input pin to be used to enable GPIO Test Mode, and to transition through the test stages.

When asserted for two seconds at power on, GPIO Test Mode is entered. The pin must have been configured as an input before this command (see the gpio commands, above).

`gpio <pin> test`

Designates an output pin to be tested during the output flash and output on test states.

The pin must have been configured as an output before this command (see the gpio commands, above).

`gpio <in-pin> test toggle|rst|set|copy <out-pin>`

Configures an input pin to cause a value change of an output pin while in GPIO Test Mode.

The output value can be set, reset, toggled, or set to the value of the input pin. The output setting is effective only on the edges of the input. The pins must have been configured as input or output pins before this command (see the gpio commands, above).

host command

`host <spi | uart>`

`host uart speed <baudrate>`

`host uart mode <mode>`

If gpio mode isn't enabled, these commands set the communication method between the module and the MCU. For UART, there is an option to set the baudrate for the communication and mode. The GPIO command overrides this.

The <mode> option can be used to set the parity for module-MCU communication. By default, the configuration is set to odd parity, which is mode 1. The other acceptable values of mode are 0 for no parity and 2 for even parity. This feature is supported from module release **2.4-beta**.

hw command

`hw [rtc_src [lsi | lse]]`

Configures various platform-dependent settings.

The rtc_src selects either an internal R/C oscillator or an external crystal/oscillator. This is settable only in setup mode.

id command

`id <token> <value>`

Example: id serial "AC000W000123456" will set the serial number in the configuration. The possible tokens are:

`serial`

Sets Ayla serial number (e.g., AC000W000123456)

`model`

Sets Ayla model number (e.g, AY001MTP1)

`mfg_model`

Sets Manufacturer's model number, (e.g., WD101)

`mfg_serial`

Sets Manufacturer's serial number.

`dev_id`

Sets Ayla serial number (usually same as serial).

`mac_addr`

Sets primary MAC address for device (e.g.,"02:02:03:04:05:06" (colons are optional)). It sets the ID token in the running configuration. A "save" command must be used to make this effective after reboot.

img command (4390 only)

```
img [boot <slot>] [erase <slot>]
```

Without parameters, shows all images that are present.

`'img boot <slot>'`

Gets module to try booting to image in the slot next time it restarts.

`'img erase <slot>'`

Erases image in the given slot. You can only erase images, which are not currently selected as active image, or have not been marked as last known good.

locale command (OBSOLETE)

The locale command is used to configure local server page strings.

```
locale [--uri] [<str #> "<string>"]+
```

Changes local string #. Optional uri flag if str is uri-encoded

```
locale reset
```

Resets all the locale settings to the default

```
locale use [default|configured]
```

Switches from using the default or configured settings

log command

Modules <module_name>	Log Levels
• client – manages device-to-service and mobile LAN Agent processes • conf – manages configuration details • dnss – manages DNS & mDNS server	• error (rows with 'e') – logs severe or critical errors • warning (rows with 'w') – logs conditions which do not cause device failure, but need to be reported

Modules <module_name>	Log Levels
processes • netsim – (not supported) • notify – manages notifications • server – manages internal web server processes • wifi – manages wifi processes • ssl – manages secure sockets layer • log-client – manages client logging process • io – manages input/output processes • sched – manages schedules • eth – manages ethernet processes	• pass/fail – logs results of manufacturing and self-tests • info (rows with 'i') - logs messages related to normal operating conditions • debug (rows with 'd') – logs more detailed diagnostic events (not recommended for production devices) • debug2 (rows with 'd') logs additional verbose details than 'debug' level (not recommended for production devices) • metric – logs performance statistics. Metrics like how long https connections to the cloud were taking. Relies on logging service. (deprecated) • all – enables all log levels • none – disables all log levels. **Note**: Includes the default above mentioned 'info' ('i') and 'warning' ('w') level.

```
log [--mod <module_name>] [+|-][pass|fail|info|debug|debug2|metric|all|none]
```

Controls what types of events are logged and how.

```
log debug
```

Turns on debug level logging for all the modules.

```
log mod wifi -info
```

Turns off info level logging for wifi module.

```
log all
```

Turns on all levels of logging for all the modules.

```
log --mod wifi"
```

Displays the current logging settings for wifi module.

```
log
```

Displays the current logging settings for all the modules.

log-client command

The log-client command is used to enable or disable the log-client.

```
log-client
```

Shows the current status of the log-client

`log-client enable`

Enables the log-client if it was disabled

`log-client disable`

Disables the log-client if it was enabled

log-snap command

This command is new in **bc-1.7**.

`log-snap [<1..n> | erase | list | save]`

With no further options, log-snap will show the count of snapshots saved and approximately how many more snapshots will fit in the remaining space.

With a numerical argument, shows the corresponding log snapshot contents.

`log-snap <save>`

Causes the last portion of the log to be saved to a snapshot.

`log-snap <list>`

Lists each snapshot and it's save time (if known).

`log-snap <erase>`

Clears all snapshots.

mfg_mode command

`mfg_mode <disable>`

Sets the mode which should be saved by the next "save" command by disabling manufacturing mode.

`mfg_mode <save>`

The running configuration is also changed, however, the CLI command "save" will operate as if in manufacturing mode anyway until the next reboot.

mfi command

`mfi [enable|disable][bsi <bundle_seed_id>]`

`[proto <external_protocol_string>][addr <i2c_addr>]`

Without parameters, shows the current MFI configuration.

`'mfi enable'`

Enables MFI support (if HW present).

`'mfi disable'`

Disables MFI support.

`'mfi bsi <bundle_seed_id>'`

Configures IOS application bundle seed ID. This will be passed to IOS device during MFI WAC configuration.

`'mfi proto <external_protocol_string>'`

Configures the protocol string which will be passed to IOS device during MFI WAC configuration.

`'mfi addr <i2c_addr>'`

Sets the I2C address of the MFI chip. This can either be 0x20 or 0x22, depending on wiring.

notify command

Syntax (prior to **1.5.3**, and in **1.6**):

`notify poll_interval <# of secs> (Note: poll_interval not configurable after bc-1.5.2)`

`"notify poll_interval <# of secs>> -`

Sets how often module will check for reachability to the Ayla Service (internet connectivity check). Default is 5 minutes.

Syntax: (in and after **1.5.3**, after **1.6**):

`notify`

Shows the notify poll_interval and, on internal builds, the state of the notifier.

oem command

`oem <name>`

`oem model <model-name>`

The oem command configures the manufacturer's name and model number of the device that uses the Ayla module. This requires mfg_mode or setup_mode.

`oem key <oem-secret>`

The OEM secret is a per-OEM string that is assigned by Ayla when the OEM ID is assigned. It is not stored by itself in the module, but instead an encrypted version of the OEM-ID and OEM-model and this secret are formed and stored. Whenever the OEM name or model is changed, this encrypted stored string is cleared, and the OEM secret must be re-entered.

The stored encrypted info is sent to the service to verify that the device was made by the OEM and should have the appropriate device template and access permissions.

`oem host_version <version>`

The oem command can also be used in GPIO mode to set the host_version string for template association.

power command

```
power <mode|current> <max_perf|default|min|standby>
power <awake_time|standby_powered|unconf_powered> <time_in_seconds>
```

Controls how power management behaves.

"power current"

Adjusts the currently active power savings level,

"power mode"

Controls the power savings level that the module will be in when it boots up next time.

"power awake_time"

Controls how long module will stay in default power savings mode after activity.

After this time module goes back to min power savings mode. This timer is used only when power savings mode is either min or standby.

"power standby_powered"

Controls how long module stays active after booting up.

After timer fires, module CPU goes to standby low-power mode. This timer is used when power savings mode is set to standby.

"power unconf_powered"

Controls how long module stays active after booting up when there is no valid Wi-Fi profile present.

This timer is used when power savings mode is set to min or standby, and there is no configured WIFI profile.

reset command

```
reset [factory]
```

Immediately resets the module.

If the "factory" argument is given, the module copies the factory configuration to the startup configuration and then restarts.

rssi command

```
rssi
```

Starts calculating the average RSSI over the next couple of seconds and then displays the RSSI as well as the Wi-Fi antenna selections.

run command

```
run <program> [<new-version>]
```

Reboots into the specified program.

Currently only "flup", "mod", and "patcher" are supported out of which only "flup" is supported for release builds. When "flup" is used, in version **1.3** and later, the new version must be supplied for compatibility checking.

save command

```
save
```

Saves the running configuration to the startup configuration.

If the system was booted in setup mode, the running configuration will also be saved to the factory configuration.

sched command

```
sched <schedule #> name <schedule name>
```

Configures the name of a schedule.

Only applicable if module is configured to be in gpio mode. For example: `sched 0 name daytime_sched` changes the name of schedule 0 to "daytime_sched".

secure command

This feature was introduced in version **2.6** and requires that the module was manufactured using **2.6** production package or later, with booter version being **1.2** or later. This command can only be used in setup or manufacturing mode. This is supported only for STM32-based devices.

```
secure
```

If the secure command is given without arguments, shows the current status of the feature.

```
secure [boot|flash] enable
```

The secure boot enable command enables signature checking on the module.

Once enabled, the booter will verify whether the current module image, patcher and flup image are signed or not on each boot. It also enables signing checks in the booter by writing the internal flash (STM32-based).

The secure flash enable command enables flash read-out protection on the STM32-based devices.

These provisions cannot be disabled except by reflashing the entire device in the case of STM32-based modules.

server command

```
server prop time_limit <secs>
```

(only used in setup mode) Enables an HTTP client on the local LAN to access the properties on a connected MCU, via the module's HTTP server, for a limited time. It is intended for device testing during manufacturing.

For this command to take effect, the Wi-Fi module must be in AP Mode, and must not yet be registered with a user.

The time-limit value is in seconds, and must be no greater than 600 (i.e., 10 minutes). Access is automatically disabled after the time limit has elapsed. Setting the time limit to 0 disables access.

```
server security [enable | disable]
```

(only used in setup mode) It is supported from version 2.6 onwards. This is recommended to enhance security of the module in AP mode.

If disabled, the module allows wifi setup through clear-text server requests and through wifi javascript page. However, once enabled clear-text access is blocked and wifi setup can be done only via an encrypted session through the mobile app. Wifi javascript page is no longer accessible.

setup_mode command

```
setup_mode disable
```

Sets the mode, which should be saved by the next "save" command by disabling setup mode.

Certain parameters (TBD) will not be changeable except in manufacturing or setup mode.

show command

```
show conf
```

Shows the current running configuration.

Long fields are not completely shown. This is for diagnostic information, and is not in a format that can be used to configure the device.

```
show id
```

Shows the current ID configuration, including model, serial, dev_id, mfg_mode, mfg_serial, stm32 signature, and MAC address.

```
show oem
```

Shows the oem name and model information.

```
show host
```

Shows some statistics about the interface to the host mcu.

```
show test_status
```

Shows test_status' shows a non-zero number if manufacturing tests have been completed.

The number is the time in seconds since 1970.

`show version`

Shows the module firmware version.

`show wifi`

Shows the current Wi-Fi status, including the AP it is connected to, if any, and the IP address. It then lists the enabled profiles and the scan list.

time command

`time`

The time command displays the time information of the module. For example:

```
UTC Time:   04/10/2013 19:05:35, Time Since Boot: 254353 ms
Local Time: 04/10/2013 11:05:35, Timezone: 480 mins west of UTC
DST Active: 0, DST Offset = 0 mins
```

version command

This command is new in version **2.6**.

`version [mod | boot | all]`

Show the installed version of firmware components.

This command, without arguments, is the same as the "show version" command, showing the module firmware version only.

wifi command

Some Wi-Fi commands may not be acted on until a "wifi commit" command. This allows a "save" to be issued before the Wi-Fi changes are made.

`wifi ant <0|1|3>`

Sets the antenna to be used for the next join or AP mode. 1 maps to connector J1, 0 maps to J2, 3 indicates both antennas may be used.

`wifi ap_mode conditional <0|1>`

Sets the device's AP mode to be conditional (1) or not (0).

When ap_mode is not conditional, the device goes into AP mode when it cannot find a Wi-Fi network to connect for any of the 10 profiles.

When AP mode is conditional, the device will only go into AP mode when no profiles are present. This is more secure because it requires a factory reset or other method of deleting the profile before re-entering Wi-Fi setup.

`wifi commit`

Causes the Wi-Fi manager to act on the changes made by other wifi CLI commands. For example, if "wifi enable 1" is used, the actual join or AP mode won't occur until "wifi commit" is used.

`wifi disable`

Immediately turns off Wi-Fi.

`wifi enable [<1|0>]`

Enables or disables Wi-Fi.

Without arguments, this immediately turns on Wi-Fi. With a 0 or 1 argument, it changes the configuration without effecting the state of the Wi-Fi interface until the next commit.

`wifi join`

Tries to connect to the current profile

`wifi key <password>`

Sets the Wi-Fi security key for the current profile.

`wifi listen <dtim_in_beacons>`

Sets the Wi-Fi listen interval in number of beacons. Not used in max_perf power mode. By default this is set to 3.

`wifi mac_addr <mac>`

Sets the station MAC address to be used as a station or in AP mode.

`wifi power <db>`

Sets the maximum transmit power, in db, of the Wi-Fi radio.

Setting the db to 0 will revert the transmit power value to the default setting for the module. The current transmit power setting is shown in the output of the "show wifi" command.

Currently, this command is only supported on Broadcom-based modules.

`wifi scan`

Scan the network for available AP.

`wifi security <type>`

Sets the Wi-Fi security for the current profile. <type> may be WPA2_Personal, WPA, WEP, or none.

Currently, these entries are case-sensitive.

`wifi ssid <name>`

Sets the Wi-Fi SSID for the current profile.

`wifi ssid-mac <name>`

Sets the Wi-Fi SSID for the current profile after appending the MAC address from the ID to <name>.

On most modules, the Wi-Fi chip must be enabled before the MAC can be gotten from the Wi-Fi chip, so this is best done after enabling. This is useful only for the 'ap' profile. For example, if the MAC address is 12:34:56:78:9a:bc, the command 'wifi ssid-mac Ayla-' will set the SSID to "Ayla-123456789abc'.

```
wifi profile <0-9 | ap | enable | disable | erase>
```

With values 0 to 9 or ap, sets the Wi-Fi profile to be acted on by the other Wi-Fi commands.

The "ap" profile sets the parameters to be used when the module is in access-point mode. The default profile is 0.

With "enable" or "disable", this enables or disables the selected profile.

With "erase" this removes configuration for selected profile.

```
wifi save_on_ap_connect <0|1>
```

Clears or sets the behavior of the Wi-Fi manager to save selected new profiles chosen with the internal web server or SPI interface when the profile is used to successfully associate with an access point and a DHCP address is acquired. Defaults to 0.

```
wifi save_on_server_connect <0|1>
```

Clears or sets the behavior of the Wi-Fi manager to save selected new profiles chosen with the internal web server or SPI interface when the profile is used to connect to the configured device service.

This defaults to 0. But is set to 1 by ayla_setup during manufacturing.

If save_on_server_connect is set to 1 and no device server is configured, the behavior is as if save_on_ap_connect is set to 1.

```
wifi setup_ios_app <name_of_the_app>
```

Module can suggest which iPhone application to fire up when the IOS device connects to it.

Can only happen when module is in AP mode. If name_of_the_app is set to empty string, module does not suggest any string, and iPhone will only stay connected to it for a little while. Defaults to empty string.

```
wifi setup_mode airkiss <enable|disable>
```

To enable/disable WeChat's Airkiss wireless setup protocol. This option is present in **bc-1.12** and later

```
wifi setup_mode airkiss key <16bit_AES_key>
```

Specify a 16bit_AES_key for Airkiss decryption.

This is the key which will used by an Airkiss client to encrypt the Wi-Fi configuration information during Airkiss Wi-Fi setup. The key needs to be input in hexadecimal. This option is present in bc-1.12 onwards.

```
wifi setup_mode airkiss vid <WeChat_OEM_id>
```

Sets up the OEMs WeChat ID on the device which is advertised during WeChat Airkiss2.0 LAN Disocovery. This option is present in **bc-2.5** onwards.

```
wifi wps
```

Initiates Wi-Fi Protected Setup Button Press Configuration.

AP's WPS button has to be pressed as well, and then AP will allow module to join it without entering password manually. Time window for this configuration to complete is less than 2 minutes.

```
wifi region <region_code>
```

Sets the geographical region in which the Wi-Fi radio will operate. This setting restricts the channels, protocol, and RF behavior of the module according to the regulations of the region specified. The setting can only be changed in manufacturing or setup mode. This option is present in **bc-2.5.2** onwards.

Valid values include:

Value	Country or Region
""	The empty string resets the region to the default value for the module
US	United States
CA	Canada
EU	The European Union
JP	Japan (not supported on 4390)
CN	China
AU	Australia

Appendix A

A.1 I/O types

The number and kind of GPIOs will vary between different module types since the module vendors bring out different GPIOs, but all modules will provide some digital inputs and outputs.

Each I/O can be associated with a configurable property name.

There are several I/O types possible, but to keep things as simple as possible, the following standard types will be supported:

- **Digital Outputs:** These are push-pull or open-drain digital I/Os. The associated to-device property is Boolean. The pin can be configured for active-high so that the property is 1 when the input is high and 0 when the input is low, or for active-low (inverted).
- **Digital Inputs:** These are floating inputs with an optional internal pull-up or pull-down.
- The associated from-device property is boolean. A new datapoint is sent up only if the input is different from the previous value.
- **Analog Outputs:** [TBD] These are analog outputs and will be driven to a voltage between 0 and the maximum when the property value is set between 0 and 1000.
- **Analog Inputs:** [TBD]. These are analog inputs and associated integer properties. The integer property will have a value from -1000 to +1000 depending on the voltage input from the lowest negative range (if supported) to the highest positive range possible. The details of the voltage range depend on the voltage supplied to the module and the specific module specifications. See the sampling section for a description on how inputs are sampled.

A.2 Sampling

Inputs may have an associated sample configuration that indicates how the input should be sampled. The options are:

- **default** (button): the input is edge-detected and debounced with a compiled-in 10 ms debounce time. The debouncing detects changes in the input but acts on them only after the input is stable for at least 10 ms and the input is different from the previous stable state.
- **edge**: [TBD] every edge on the input is sent to the service.
- **sampled**: [TBD] the input is sampled periodically at a configured rate.

The enabled inputs will be sampled and debounced according to compiled-in constants for now.

A.3 Other I/Os

When in GPIO mode, the I/Os SPI lines, WKUP, and several other I/Os become assigned to GPIO properties. The LINK_N I/O function is unaffected by GPIO mode. The READY_N I/O becomes unasserted so that if GPIO mode is enabled on a device with an MCU connected, the MCU will not see the SPI bus as ready.

A.4 Associate an input pin with an output pin

A digital input can be configured to be used as a button to toggle a digital output. The digital output property change is "echoed" to ADS. The input may or may not have a property of its own.

- Gpio - show all GPIO settings
- gpio <pin> - show settings for one pin
- gpio <property> - show assignment for one property

A.5 GPIO Configuration for SmartPlug Demo on USI BM-09 EVB AY001MUV

```
gpio enable
gpio pc2 prop Green_LED inv od
gpio pc3 prop Blue_LED inv od persist
gpio pa4 prop Blue_button inv pu toggle pc3
gpio pa6 od inv
gpio pb7 od inv
gpio status wifi pa6
gpio status service pb7
gpio commit
save
reset
```

The Wi-Fi status output on pa6 here will usually go to an amber LED on the SmartPlug using a different GPIO. The service status output on pb7 here will usually go to a green LED on the SmartPlug using a different GPIO. See the Smartplug Requirements document for details.

The GPIO commit isn't needed unless GPIOs were previously enabled.

A.6 Restrictions

On modules based on the STM32, it is currently not possible to use two pins in different banks but the same number (e.g., pa1 and pb1) both as inputs. This is not enforced by the code, but will not work. The device designer must be aware of this.

4250 Burton Drive, Santa Clara, CA 95054
Phone: +1 408 830 9844
Fax: +1 408 716 2621