AIRTOPIA
ENHANCED CLIMATE CONTROL

API GUIDE

MODBUS TCP/IP Commands
Version 1.3

Contents

TCP Commands

Introduction

The Airtopia controller can be used on an internal Local Area Network (LAN), without Internet connection. The controller can be managed directly via a Modbus TCP/IP connection from a local BMS. Connecting directly over a LAN is perfect for security conscious organisations that do not want an uplink to the Internet. We recommend that you deploy units in a private Virtual Local Area Network (VLAN) unconnected from other computers, the Internet and so on.

Objective

This document describes the Airtopia control system's Modbus protocol. The protocol enables a network master to modify settings and receive measurements from the Airtopia controller using a TCP/IP over an Ethernet connection. This document provides a description of each command attribute, its function, the command attribute key and the value for each command. Each command table includes an example to assist the programmer with correct syntax when designing their interface.

System Description

The Airtopia control system provides a translation layer between numerous static interfaces (CBus, ModBus, JSON/TCP, the website) and many models of Split System Air Conditioner. This means that a single control program on a BMS can be used to control many varieties of Air Conditioner, without large change to the underlying code. The fields available for each Air Conditioner varies, as each AC may have different features.

EXAMPLE: the ability to turn the unit on/off, set a temperature setpoint and select a mode are probably common to all ACs; whereas features like Horizontal/Vertical swing, Powerful Mode, etc are AC specific. The system utilises a secure Internet based commissioning system, which allows for easy setup of units, without port forwarding, or determining which IP the unit is using.

Modbus Port Number

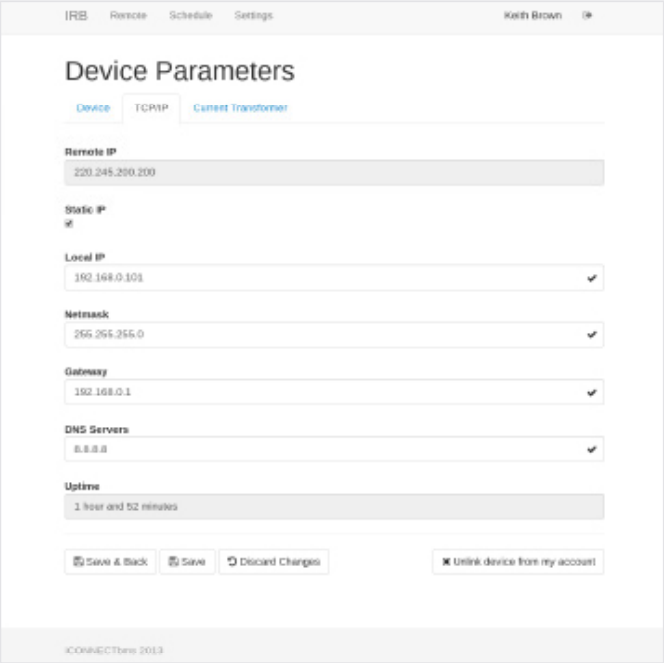
The Airtopia controller accepts Modbus connections on port 502.

TCP connection timeout

The controller will drop any connection which hasn't seen packet flow in the last 3600 seconds (one hour). To avoid this, either an empty JSON object ('{}') or a single NULL ('\0') byte can be used as keepalive. Out of band keepalives are not recommended as network equipment is not obliged to pass them on. The Airtopia controller will not attempt to keep a connection alive.

Commissioning

Before the unit can be used in an offline setting, it must be commissioned using the Airtopia Cloud service. Attach the device to an Internet connected network and use the web portal to select the appropriate Air Conditioner profile and static IP settings (if required). We recommend setting the IP to static in a BMS situation. The Current Transformer Ratio, if optioned, can also be set here (typically 1000:1 or 3000:1). To configure a static IP, navigate to http://app.airtopia.global/ and log in. Select 'Settings' from the top toolbar, select 'My Devices' followed by the device that you wish to configure. Under the 'TCP/IP' tab there are a list of fields that you must fill out correctly. First tick the 'Static IP' checkbox. The fields below will become editable. Here is an example:



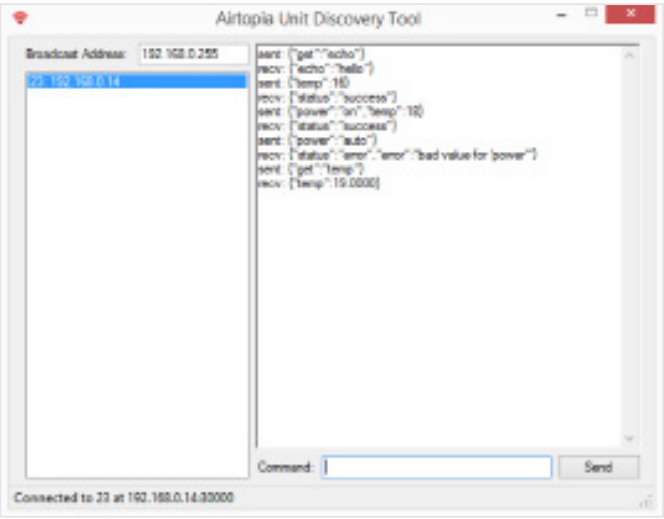
Once complete, press save and wait for each field to change from ↻ to a ✓

Note that when the static IP settings do not allow for connection via the Internet, the Network Reset on the unit can be pressed. This will force the unit back to DHCP, and allow for reconfiguration via the Airtopia Cloud.

Discovering a units IP

Now you can deploy the unit onto the private LAN. The following method of discovering IPs is used when the unit is set to DHCP and the IP is unknown. If you have selected static IP, you may skip to the next step (although this method can validate static IPs).

Using our Airtopia discovery application



You can download the Airtopia Discovery Tool from the Docs & FAQS section (http://airtopia.global/docs-faqs). Open up the discovery application and wait for the list on the left to be populated by Airtopia devices. This typically takes around 10 seconds. If you are having issues, please verify that the broadcast address is correct.

Click on the device to send messages to it. See the JSON commands below.

Using command line tools

To find the IP of the unit we must port scan the network. We can use a tool called 'Nmap (http://nmap.org/download.html#windows)'. Open up a command prompt and type:

```
nmap p30000
open
192.168.1.1/24
```

Where the IP address is an address in the local subnet. The '/24' shows the network mask, expressed as the number of ones (255.255.255.0 is typical, and has 24 binary ones). 'ipconfig' on Windows can tell you what subnet you're using.

EXAMPLE:

```
$ nmap p30000
open
192.168.0.1/24
Starting Nmap 6.46 ( http://nmap.org ) at 20140427
16:45 AUS Eastern Standard
Time
Nmap scan report for 192.168.0.2
Host is up (0.00088s latency).
PORT STATE SERVICE
30000/tcp open unknown
MAC Address: 00:04:A3:95:20:C3 (Microchip Technology)
Nmap done: 256 IP addresses (15 hosts up) scanned in 2.56 seconds
```

This shows that the device at 192.168.0.2 is listening on TCP port 30000. This is what we connect to.
The MAC address is unique for each Airtopia unit, and can be used to identify units.

Making a TCP connection

For debugging purposes we use a terminal program called 'ncat', included in the Nmap suite, you may also use 'PuTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)'. This is the stage that your BMS will make a socket connection to the controller.

EXAMPLE: if the device was on IP 192.168.0.2

```
$ ncat 192.168.0.2 30000
```

Will connect but you will be greeted with nothing! If it does not quit then you have successfully connected.

NOTE: Ncat is line buffered, so every time you hit the enter key, the packet is transmitted and the device tries to execute the command.

Sending commands

Once a connection is established, commands can be sent. The controller accepts commands in the form of a JSON (http://www.json.org/) message. JSON is a schema based on name/value pairs.

EXAMPLE: a JSON compliant object to set setpoint to 23, mode to cool and turn power on would be:

```
{"setpoint":23,"mode":"cool","power":"on"}
```

If the message was successful the controller will reply with the actual message sent (The message may go through some processing due to protocol conditions). It will return the entire state of the virtual remote.

EXAMPLE:

```
Send: {“mode”:”cool”,”setpoint”:22}
Recv: {“power”:”on”,”mode”:”cool”,”hswing”:”auto”,”fan”:”4”,”setpoint”:”20”,”vswi
ng”:”auto”,”econo”:”off”,”powerful”:”off”,”comfort”:”on”,”sensor”:”off”,”quiet”:”
on”}
```

NOTE: Commands must be sent with a trailing new line character (‘\n’). An unsuccessful message will be replied with an error message such as:

```
Send: {“mode”:
Recv: {“error”:”JSON incomplete”}
Explanation: No closing brace; invalid JSON object.
Send: {quiet:on}
Recv: {“status”:”error”,”error”:”No such command ‘quiet’ in the protocol definition”}
Explanation: The AC profile loaded doesn’t support the ‘quiet’ mode.
```

Sensor data query

Sensor information can be requested from the controller using one of the following commands:

NOTE: It is not possible to issue a ‘get’ command and set a AC specific command in the same JSON packet. They must be sent as individual commands.

```
Send: {“get”:”temp”}
Recv: {“temp”:22.4}
```

To retrieve current in Amps from the current transformer clamp:

```
Send: {"get":"current"}
Recv: {“current”:0.6}
```

To retrieve the logic state of the closed contact switch:

v

```
Send: {“get”:”logic”}
Recv: {“logic”:1}
```

You can request all inputs in one go:

```
Send: {“get”:”inputs”}
Recv: {“temp”:24.15,”current”:0.00,”logic”:0}
```

To retrieve the current state of the internal air conditioner settings:

```
Send: {“get”:”state”}
Recv: {“power”:”on”,”setpoint”:23,”mode”:”cool”,”fan”:”auto”,”quiet”:”off”,”powerful”:”off”}
```

To retrieve the loaded protocol details:

NEW: The “Protocol” command provides a simply method to return the Manufacturer and Model of the IR handheld remote loaded in the device.

```
Send: {“get”:”protocol”}
Recv: {“protocol”:”Mitsubishi RKW502A200”,”id”:234}
```

To retrieve a list of available commands for the current air conditioner:

```
Send: {“get”:”commands”}
Recv: {“power”:[“on”,”off”],”mode”:[“cool”,”heat”,”auto”,”dry”,”vent”],”fan”:[“1”
,”2”,”3”,”4”,”auto”],”vswing”:[“0”,”1”,”2”,”3”,”4”,”off”,”auto”],”hswing”:[“right
“,”mid_right”,”mid”,”mid_left”,”left”,”outward”,”off”,”inward”,”auto”],”setpoint”
:[14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33]}
```

Example Modbus Air-conditioner Commands
Slave ID: 1
Function: 03 Read Holding Register (4x)
Address: 0
Quantity: 6
Scan Rate: 5000ms

Power: 40001

Icon	Value	Comment
	1	Power on
	0	Power off

FORMAT

Write Function
Register 0
Int16 Unsigned
Big Endian

EXAMPLE:

Slave ID=1 Alias=Power Address=0 Value=1
Use Function 06=write single register

Turns the AC on

Set Point: 40002

Temperature in degrees Celsius:
- Supports a resolution of 1
- Has a minimum value of 14
- Has a maximum value of 34
- Values outside this range will be clipped

FORMAT

Write Function
Register 1
Int16 Unsigned
Big Endian

EXAMPLE:

Slave ID=1 Alias=Set Point Address=1 Value=22
Use Function 06=write single register

Changes the setpoint to 22 degrees Celsius

Mode: 40003

Icon	Value	Comment
	1	auto
	2	cool-
	3	heat-
	4	fan
	5	dry

FORMAT

Write Register
Register 2
Int16 Unsigned
Big Endian

EXAMPLE:

Slave ID=1 Alias=Mode Address=2 Value=1
Use Function 06=write single register

Changes the mode to heat

Fan: 40004

FORMAT

Write Register
Register 3
Int16 Unsigned
Big Endian

Icon	Value	Comment
	255	auto
	0-10	20%
	11-30	40%
	31-50	60%
	51-70	80%
	71-90	100%

EXAMPLE:

Slave ID=1 Alias=Fan Address=3 Value=30
Use Function 06=write single register

Changes the fan setting to the 40% fan power

Vertical Swing: 40005

Icon	Value	Comment
	1	swing between 0° and 90°
	0	default rest position

FORMAT

Write Register
Register 4
Int16 Unsigned
Big Endian

EXAMPLE:

Slave ID=1 Alias=Vswing Address=4 Value=1
Use Function 06=write single register

Changes the vertical swing setting to swing between 0° and 90°

Horizontal Swing: 40006

Icon	Value	Comment
	1	swing between left and right
	0	default rest position

FORMAT

Write Register
Register 5
Int16 Unsigned
Big Endian

EXAMPLE:

Slave ID=1 Alias=hswing Address=5 Value=1
Use Function 06=write single register

Changes the horizontal swing setting to swing between left and right mode

Example Modbus Sensor & Input Queries
Slave ID: 1
Function: 04 Read Input Registers (3x)
Address: 0
Quantity: 8
Scan Rate: 2000ms

Protocol ID:
30001 - 30002

Icon	Value	Comment
		-
		-

EXAMPLE: 255

The protocol ID that is currently loaded
Unsigned 32-bit int that can be used to verify available commands.

Use the JSON command to retreive deatiled command set. see TCP/IP Application note

Temperature Sensor:
30003 - 30004

Icon	Value	Comment
		-

FORMAT

Read Register
Register 2
Int16 Unsigned
Big Endian
Scale Factor divide by 256
Degrees Celsius

EXAMPLE: 0x2281

Raw measured temperature from the one-wire interface (5100DTS-SM) Left aligned in big-endian.
Temperature in Celsius in the upper byte
and binary fractions in the lower byte

Room Temperature 22.81 °C

Current Sensor:
30005 - 30006

FORMAT

Read Register
Register 4
Int16 Unsigned
Big Endian
Scale Factor Multiply by 10
Milliampres

Icon	Value	Comment
		-
		-

EXAMPLE: BCD 0x0100

Raw measured current from the CT interface
Left aligned in big-endian.
Current in milliamps in the upper byte, and binary fractions in the lower byte

100mA

Dry Contact:
30007 - 30008

FORMAT

Read Register
Register 6
Int16 Unsigned
Big Endian

Icon	Value	Comment
	0	open-
	1	closed

EXAMPLE: on Event run schedule

Dry contact input used to trigger an event. Motion Sensors, PIRs
Closed Contact from an Alarm system etc
Air Conditioners Fault Relay

Spare

Icon	Value	Comment
		-
		-

EXAMPLE:

Airtopia™
Level 2, 75 River Street
Richmond VIC 3121 Australia
PO Box 520
Bulleen VIC 3105 Australia
ABN 41 146 106 998

Phone: 1800 247 867
info@airtopia.com
www.airtopia.com

iConnectBMS Pty Ltd trading as Airtopia™