AirControl v1.4 API

For use with development build:

- Open the "About" dialog. Press CTRL-SHIFT-U
- Change update site URL to: http://www.ubnt.com/downloads/aircontrol/upd138/
- Click "Check for Updates" and then update to the new version shown

(URL above is not for entry into the browser address bar but into the AirControl update dialog.)

To revert to previous version:

- stop server
- Go into the <ACDIR>/webapps directory
- delete ROOT/
- delete ROOT.war
- rename previous archive to ROOT.war
- start server

External API Usage

AirControl exposes an experimental, limited web service to integrate with custom management solutions or other front-end applications. It can be used with command line tools like wget and curl or any scripting language that understands HTTP(S), including PHP, Python, HTML/JavaScript. You don't need to be a programmer to use the API, but should be ready to use the shell/command line (wget and curl are also available for Windows).

This API follows REST conventions/style and requires authentication. For modifying operations, administrative privileges are required. Those need to be protected and should only be used from a server or in a way that does not expose them to other users (for example through JavaScript in a customer facing portal etc.).

Not all HTTP clients (browsers) support HTTP PUT or DELETE methods. For those methods server side usage with an appropriate capable HTTP client is assumed.

Examples below are using **wget** utility (wget is available on all platforms, including Windows as part of Cygwin, an alternative is curl that supports the use cases below, just through different options).

Retrieving Server Info

Get AirControl status (cuurently only version):

wget "http://localhost:9080/api/system/status?auth.user=ubnt&auth.password=ubnt"

Get list of defined device groups:

wget "http://localhost:9080/api/system/devicegroups?auth.user=ubnt&auth.password=ubnt"

Retrieving Device Data

Get a device list by specifying a group name. Groups can be any custom group under "Watch" or "All Devices" (but not the auto-groupings). This list can be retrieved periodically and cached in a front-end system or consumed directly and fetched on demand.

The columns and order are as defined in the device list customizer within the AirControl GUI. So you can setup a dedicated account and customize the device table for what you want to get through the API.

Access to data through HTTP, all parameters including authentication in URL, example:

http://localhost:9080/api/devicegroup/All%20Devices?auth.user=ubnt&auth.password=ubnt

Get "All Devices" data in TSV format:

wget --header="Accept:text/tab-separated-values" "http://localhost:9080/api/devicegroup/All%20Devices?auth.user=ubnt&auth.password=ubnt"

Get "All Devices" in JSON format:

wget --header="Accept:application/json" "http://localhost:9080/api/devicegroup/All%20Devices?auth.user=ubnt&auth.password=ubnt"

There is also a sample HTML/JavaScript example for JSON and dynamic script tags that you can get from the running server: http://localhost:9080/ApiGetDeviceList.html

Access Current Statistics for single Device:

http://localhost:9080/api/devices/00:15:6D:72:42:0E/currentStats?auth.user=ubnt&auth.password=ubnt&type=tsv URL query parameters including output options are same as for device group list.

Statistics History:

Single Device:

```
http://localhost:9080/api/devices/00:15:6D:72:42:0E/stats/wlanTxBytes?
auth.user=ubnt&auth.password=ubnt&statsIntervalPeriod=hour&statsIntervalCount=24
```

Parameters: MAC ID and attribute name in the URI. Attribute name (not title) as found in the Attribute Configuration dialog in the GUI.

statsIntervalPeriod: Aggregation/time interval ('minute', 'hour', 'day', 'week')

statsIntervalCount: Number of intervals in the ouput.

Currently only supports TSV output format. The report will have a column for each interval, with the interval start date as header.

Report currently does not contain line or column summary (sum, min, max, avg).

Device Group:

```
http://localhost:9080/api/devicegroup/All%20Devices/stats/wlanTxBytes?
auth.user=ubnt&auth.password=ubnt&statsIntervalPeriod=hour&statsIntervalCount=24
```

One of the use cases is to generate usage reports and load the data into a spreadsheet application for further processing.

Get Device Map KML

http://localhost:9080/api/devicegroup/All%20Devices/map/kml?auth.user=ubnt&auth.password=ubnt

The resulting KML file can be loaded into Google earth or viewed through Google maps (http://econym.org.uk/gmap/kml.htm)

Adding and deleting devices

wget -v -O rsp.txt --post-file api-add-device.txt --header="Content-Type: application/json" "http://localhost:9080/api/devices? auth.user=ubnt&auth.password=ubnt"

The JSON data for this request:

```
"hwaddr": "00:15:6D:16:C3:00",
"ipaddr": "0.0.0.0",
"name": "api test device",
"sshPort": null,
```

```
"webPort" : null,

"latitude" : 37.41091864282257,

"longitude" : -121.9424858063698
```

Specifying the the port numbers as null or omitting them will indicate that the system wide settings should be effective for the new device.

Remove Device

curl -i -H "Accept: application/json" -X DELETE http://localhost:9080/api/devices/00:15:6D:16:C3:00?auth.user=ubnt&auth.password=ubnt"

Control Operations

A few control operations are now available: Connect, upgrade (optional for non-managed device), restore configuration. Affected devices can be specified as MAC address list or device group. Example on how to submit a request specified in a file as HTTP POST request:

```
wget -v -0 ac-response.txt --post-file ac-request.txt --header="Content-Type: application/json"
'http://localhost:9080/api/task?auth.user=ubnt&auth.password=ubnt'
```

Above submits the file api-request.txt (JSON format) to AirControl, running on localhost:9080, using login ubnt/ubnt. The account used to schedule tasks must have ADMIN priviledge.

The request will be executed asynchronously, look at the progress list in AC and the system log for results. Successful scheduled will result in HTTP status 201. Should the request return HTTP error 500, check <AC HOME>/logs/server.log. Most likely the JSON format was invalid.

Currently there is no way to poll the task status through the API based on the task identifier returned with HTTP response status 201; this is something that will be added in the future. At this time it is fire and forget as far as the API client is concerned.

If you received status 201, AirControl has scheduled the request and it will for a short time appear in the schedule list. Once processing, look for status in the respective device details (like any other in-progress operation in AirControl). Should processing fail, the error will be recorded in the event log.

The API request specifics are encoded in JSON format in the file. Sample payloads are given below, copy these into the text file referenced on the wget command line.

Each request has the common parameter deviceMacList or deviceGroup. Device groups can be specified as the simple name of any of the groups under "Watch" in the group tree (e.g. "All Devices" or "Low Signal") or, for convenience, also auto-groups, such as "AP-Groups/THOMAS-NSM2-UPD5", where the leading path component is the grouping name and the second component the specific attribute value.

Connect (put under management) device(s) by MAC ID.

```
"actionType" : "connect",
 "deviceMacList":["00:15:6D:72:42:0E","00:15:6D:A7:CE:7B","00:15:6D:16:C3:7B"],
 "login": "ubnt",
 "password": "ubnt"
(The device(s) must have been added in AirControl before.)
Connect using device authentication with SSH key instead of password:
 "actionType" : "connect",
 "deviceMacList":["00:15:6D:16:C3:7B"],
 "login": "ubnt",
 "sshKeyFile": "/ubnt/console/test/config/id rsa api"
Reboot Managed Devices
 "actionType" : "reboot",
 "deviceMacList":["00:15:6D:72:42:0E","00:15:6D:A7:CE:7B","00:15:6D:16:C3:7B"]
Upgrade device:
 "actionType" : "firmwareUpdate",
 "deviceMacList":["00:15:6D:16:C3:7B"],
 "firmwareVersion": "XM.ar7240.v5.3-beta5.7493.101210.2221"
Upgrade unmanaged device with explicit device authentication (SSH key):
 "actionType" : "firmwareUpdate",
 "deviceMacList":["00:15:6D:16:C3:7B"],
 "firmwareVersion": "XM.ar7240.v5.3-RC2.7694.110105.1630",
 "login": "ubnt",
 "sshKeyFile": "/ubnt/console/test/config/id rsa api"
```

Configuration backup for a group of devices:

```
{
  "actionType" : "backupConfiguration",
  "deviceGroup": "All Devices",
  "backupName" : "API-Backup",
  "backupCount" : 3
}
```

Restore latest backup for a group of devices:

```
"actionType" : "restoreConfiguration",
  "deviceGroup": "My Device Group",
  "backupName" : "On-Demand-Backup"
}
```

A not so obvious use case for this:

Backup a set of devices (from the AirControl UI, on-demand or scheduled)

Manipulate the configuration files with a custom routine (at your own risk!)

Restore manipulated files through the API to all units.

Note this **not a recommended solution** for configuration change, use **at own risk** if you know how to deal with system.cfg and want to accomplish things that cannot be accomplished through higher level tools at this time!

(Mass) Configuration

A limited set of parameters is available for configuration through the API (see list below examples). Supported firmware versions are 5.3 and 3.6. This is alpha functionality. Please always test on a radio that you can reset easily before applying changes to multiple or critical devices.

If you are running AirControl on Linux and using OpenJDK (not Sun JDK): Some OpenJDK distributions do not include the script engine that is required by the configuation API. I have seen it missing on Centos, Fedora OpenJDK etc. distributions. I have seen it present on Ubuntu 9.x and 10.x. If you see an error in the system log referring to the script engine, please either install the Sun JDK or email for instructions on how to add it to the existing OpenJDK (mention exact Linux and OpenJDK version).

When changing multiple devices in the same request, make sure to only specify devices that are managed and online. Changing many devices in a batch will synchronize the configuration change so that restart occurs at the same time (the larger the batch the more system resources are required to process it and may impact the performance of your server significantly). Batch request will be cancelled if any one device fails to update (necessary for

change of parameters that affect dependency within groups). Also make sure to test changes before applying to multiple/many units.

If changes do not require synchronized execution, specify serial execution to change one device at a time and ignore processing errors (add "executeSequential": true to the request). The default is "false" – synchronized batch processing.

```
"actionType" : "configure"
"deviceGroup": "MyDeviceGroup",
"executeSequential" : true,
"attributeValues" : {
    ...
}
```

As general guideline, keep batch sizes relatively small for synchronized changes such as wireless paramaters that requires all devices to be updated at once and use sequential processing other modifications.

The configuration changes are specified as a list of attributes, which will be applied on top of the current configuration on the device. Each request can contain multiple key-value pairs. Some of those parameters have dependencies. For example, you first need to enable WPA security before attempting to set the PSK. Likewise, you only need to send the PSK if WPA was already enabled. Below are examples only, which you can adjust to the current configuration state of the devices.

Note that the keys used, while they may match the property names in system.cfg on the device, are not the same property names. They are mapped by AirControl and you will only be able to change what is published in the API. (Amongst other reasons this is necessary to enforce integrity rules and guard from misconfiguration.)

Examples:

Change security to WPA and set PSK:

```
{
  "actionType" : "configure",
  "deviceMacList":["00:15:6D:16:C3:7B"],
  "attributeValues" : {
      "wireless.1.security" : "wpa",
      "wireless.1.security.wpa.auth" : "WPA-PSK",
      "wireless.1.security.wpa.psk" : "1234567890"
  }
}
```

```
Change security to WEP and set key:
 "actionType" : "configure",
 "deviceMacList":["00:15:6D:16:C3:7B"],
 "attributeValueList" : [
    ["wireless.1.security", "wep64"],
    ["wireless.1.security.wep.key-type", "2"],
    ["wireless.1.security.wep.key", "62345"]
Turn off security:
 "actionType" : "configure",
 "deviceMacList":["00:15:6D:16:C3:7B"],
 "attributeValueList" : [
    ["wireless.1.security", "none"]
Change SSID, wireless mode, channel width, channel shifting (v5.3, M2):
 "actionType" : "configure",
 "deviceMacList":["00:15:6D:16:C3:7B"],
 "attributeValueList" : [
    ["radio.1.mode", "stawds"],
    ["wireless.1.ssid", "SSIDTEST1"],
    ["radio.1.clksel" , "E"],
    ["radio.1.rate.max" , "7"],
    ["radio.1.chanshift", "enabled"]
```

Change admin user:

```
{
  "actionType" : "configure",
  "deviceMacList":["00:15:6D:16:C3:7B"],
  "attributeValueList" : [
        ["system.users.1.name", "admin"],
        ["system.users.1.password.confirm", "ubnt"],
        ["system.users.1.password", "ubnt"]
}

Setup ping watch dog:
{
  "actionType" : "configure",
  "deviceMacList":["00:15:6D:16:C3:7B"],
  "attributeValueList" : [
        ["service.pwdog.status", "disabled"],
        ["service.pwdog.host", "127.0.0.1"],
        ["service.pwdog.period", "30"],
        ["service.pwdog.delay", "60"],
        ["service.pwdog.retry", "3"]
     ]
}
```

List of currently supported parameters as of v1.3.7 (will be expanded in the near future and as per feedback received):

More has been added since. Use the device configuration dialog in the AirControl GUI to confirm supported settings. Make changes in the configuration tree and then review the parameters on the confirmation step. Values displayed there are the same that need to be sent through the API.

Parameter	Allowed values/description
RADIO SETTINGS	
AirMax Priority radio.1.airmax-priority	{0:'High',1:'Medium',2:'Low',3:'None'}
radio.1.mode	Wireless mode: "sta", "stawds", "ap", "apwds"
IEEE mode radio.1.ieee_mode	Available modes depend on product. FM version 3.x: {'a':'A','b':'B only','g':'B/G mixed,'pureg':'G only} FM version 5.x: {'11naht': 'A/N mixed','11nght': 'B/G/N mixed'}
Channel Width radio.1.clksel	"0" – 20 MHz

	"1" – 10 Mhz "2" – 5 Mhz "E" – Auto 20/40
Channel Shifting radio.1.chanshift	"disabled" "enabled"
Max Data Rate radio.1.rate.max	Maximum TX Rate as index. Compare drop-down in AirOS UI Wireless Tab. Values for AirOS 5.3: [7,6,5,4,3,2,1,0] and [150] for 2x2 chain
	Values for 3.6: { [70] for 5GHz / A mode } { [30] for 2GHz / B only } { [70] for 2GHz / G only } { [70] for 2GHz / B/G mixed / 5 or 10MHz } { [110] for 2GHz / B/G mixed / 20MHz }
Auto Data Rate radio.1.rate.auto.status	"disabled" "enabled"
Output Power radio.1.txpower	The output power at which wireless module transmits data . The minimun and maximum power depends on product. Below are the examples Bullet M5:- min:6dBm,max:25dBm Bullet M2:- min:12dBm,max:28dBm Rocket M2X2:- min: -3dBm,max:28dBm Picostation 5:- min:3dBm,max:22dBm PicoStation 2:- min:4dBm,max:20dBm
Ack distance radio.1.ackdistance	It should be less than max distance for given IEEE mode and channel width Defaults to max distance if input distance is greater than max distance Examples: AirOS 3.x • {[max:32.7miles] for B/G Mixed and 20Mhz}, • {[max:65.9miles] for B/G Mixed and 10Mhz},

	 {[max:132miles] for B/G Mixed and 5Mhz}, {[max:67miles] for B Only and 20Mhz}, AirOS 5.x {[max:32.2miles] for B/G/N Mixed and 20Mhz}, {[max:65.3miles] for B/G/N Mixed and 10Mhz}, {[max:131miles] for B/G/N Mixed and 5Mhz},
Ack Timeout radio.1.acktimeout	It should be between Min and Max values . Min and Max values for timeout are derived based on IEEE mode, channel width and depend on the AirOS version. If value is <min defaults="" for="" is="" it="" min="" then="" to="" value="">max defaults to max Examples:</min>
	 AirOS 3.x {[min:21 max:372] for B/G mixed and 20Mhz}, {[min:37 max:744] for B/G mixed and 10Mhz}, {[min:73 max:1489] for B/G mixed and 5Mhz}, {[min:25 max:744] for B Only and 20Mhz}, {[min:11 max:204] for A Only and 40Mhz}, AirOS 5.x {[min:27 max:204] for A/N mixed(11naht40,11naht40Plus,11naht40Minus) and 40Mhz}, {[min:37 max:409] for A/N mixed(11naht20) and 20Mhz}
Auto Ack radio.1.ack.auto.status	enabled or disabled
Muticast Rate radio.1.mcastrate	AirOS 3.x ONLY. Possible values are same as "radio.1.rate.max"
Rate Algorithm radio.1.rate_modue	Possible values AirOS 3.6: {"ath_rate_sample":"Optimistic","ath_rate_onoe":"Conservative","ath_rate_minstrel","EWMA "}
WIRELESS/SSID	
wireless.1.ssid	<essid></essid>
wireless.1.mac_acl.status	enabled or disabled

wireless.1.mac_acl.policy	"allow", "deny"
wireless.1.mac_acl.list	comma separated list of MAC IDs
wireless.1.security	"wpatkip" "wpaaes" "wpa2" "wpa2tkip" "wpa2aes" "wpa2aes" "wep64" "wep152" "none"
wireless.1.security.wpa.auth	"WPA-PSK", "WPA-EAP" station mode only (valid with "wireless.1.security"= "wpa"XXX)
wireless.1.security.wpa.psk	<psk> (valid with "wireless.1.security.wpa.auth"= "WPA-PSK")</psk>
wireless.1.security.wpa.eap.type	"TTLS", "PEAP" valid with "wireless.1.security.wpa.auth"= "WPA-EAP" station mode only
wireless.1.security.wpa.eap.inner	"MSCHAPV2" // fixed
wireless.1.security.wpa.eap.anonymous-identity	
wireless.1.security.wpa.eap.identity	
wireless.1.security.wpa.eap.passwd	
wireless.1.security.wep.key-length	WEP Key Length: "wep64" – 64 bit, "wep128" – 128 bit
wireless.1.security.wep.key-type	"1" – hex "2" - ascii

wireless.1.security.wep.key	WEP key
	(Length depends on WEP key length and type, see AirOS UI Wireless Tab)
wireless.1.security.wep.authmode	"Authentication Type": "1" - open, "2" - shared key
SYSTEM	
system.device.name	<name device="" of="" the=""></name>
system.latitude system.longitude	System location, coordinates in decimal notation, e.g122.049200
system.users.1.name	<user name=""> - default "ubnt"</user>
system.users.1.password.confirm system.users.1.password	The new password, sent twice.
system.users.2.status system.users.2.name system.users.2.password	Configuration of read-only account. Status is either "enabled" or "disabled"
SERVICES	
service.pwdog.status	PING WATCH DOG "enabled" or "disabled"
service.pwdog.host	IP Address To Ping
service.pwdog.period	Ping Interval (seconds)
service.pwdog.delay	Startup Delay (seconds)
service.pwdog.retry	Failure Count To Reboot
service.sshd.status	SSH "enabled" Or "disabled" - AirControl requires SSH enabled
service.sshd.port	
service.snmp.status	SNMP "enabled" Or "disabled"
service.snmp.community	
service.snmp.contact	
service.snmp.location	
service.ntpclient.status	NTP "enabled" Or "disabled"

service.ntpclient.server	
service.webserver.https.status	WEBSERVER "enabled" Or "disabled"
service.webserver.https.port	
service.webserver.http.port	
service.webserver.session.timeout	
service.telnet.status	TELNET "enabled" Or "disabled"
service.telnet.port	
service.syslog.status	SYSLOG "enabled" Or "disabled"
service.syslog.remote.ip	
service.syslog.remote.port	
service.syslog.remote.status	"enabled" Or "disabled"
service.dynamic-dns.status	Dynamic DNS, AirOS 5.3 only: "enabled" Or "disabled"
service.dynamic-dns.hostname	
service.dynamic-dns.username	
service.dynamic-dns.password	