MaxAir Technical – Application Programming Interface (API)

A set of APIs are available to facilitate communication between MaxAir and other application software. An example could be integration with Amazon's Echo to enable voice control of MaxAir functions.

In order that the APIs can be called without the need for the .php file extension, the Apache2 mod_rewrite module needs to be enabled. A script file 'enable_rewrite.sh' is provided to enable this feature and can be executed using the command 'bash enable_rewrite.sh'.

The API's available are:

- binarySet
- boostSet
- getController
- getTemperature
- getZoneStatus
- modeSet

The APIs are used by calling using the MaxAir systems IP address and a parameter representing a Zone Name e.g. ***http://192.168.0.100/api/binarySet?zonename=Central%20Heating***. All APIs are capable of returning data in 'JSON' format, additionally binarySet and boostSet API can pass actions to the MaxAir system.

binarySet API

Used to return the current state of a zone or set a new state for the zone.

Example 1 - get the current state of the 'Central Heating' zone:

http://192.168.0.100/api/binarySet?zonename=Central%20Heating	`{"success":true,"state":false,"state_str":"off"}`

The returned JSON contains 3 components, 'success' – either true or false depending on if the API call was successful or note, 'state' – either true or false depending on the zone's current state, 'state str' – OFF or ON depending on the 'state'.

If a zone contains multiple controllers e.g. a Lamp zone with x individual light switches, then an array will be returned, with each element structured as above.

Example 2 - set the state of the 'Central Heating' zone to 'ON':

http://192.168.0.100/api/binarySet?zonename=Central%20Heating&state=1	`{"success":true,"state":true}`

The returned JSON contains 2 components, 'success' – either true or false depending on if the API call was successful or not, 'state' – either true or false depending on the zone's new state.

boostSet API

Used to return the current boost status of a zone or set a new boost state for the zone.

Example 1 - get the current boost status of the 'Central Heating' zone:

http://192.168.0.100/api/boostSet?zonename=Central%20Heating	{"success":true,"state":false,"state_str":"off"}

The returned JSON contains 3 components, 'success' – either true or false depending on if the API call was successful or not, 'state' – either true or false depending on the zone's current boost status, 'state str' – OFF or ON depending on the 'state'.

If a zone contains multiple controllers e.g. a Lamp zone with x individual light switches, then an array will be returned, with each element structured as above.

Example 2 - set the boost state of the 'Central Heating' zone to 'ON':

http://192.168.0.100/api/boostSet?zonename=Central%20Heating&state=1	{"success":true,"state":true}

The returned JSON contains 2 components, 'success' – either true or false depending on if the API call was successful or not, 'state' – either true or false depending on the zone's new boots state.

getController API

Used to return the current ON/OFF state of the system controller.

Example :

http://192.168.0.100/api/getcontroller	{"success":true,"state":"OFF"}

The returned JSON contains 2 components, 'success' – either true or false depending on if the API call was successful or not, 'state' – the current ON/OFF state of the system controller.

getTemperature API

Used to return the current temperature value from a temperature sensor, the response depends on whether or not the temperature is mains or battery powered.

Example 1 - get the current temperature from the mains powered 'Hot Water' temperature sensor:

http://192.168.0.100/api/getTemperature?sensorname=Hot%20Water	{"success":true,"state":"29.40","datetime":"2021-03-30 18:03:31"}

The returned JSON contains 3 components, 'success' – either true or false depending on if the API call was successful or not, 'state' – the current sensor temperature value, 'datetime' the time the sensor was last updated.

Example 2 - get the current temperature from the battery powered 'Conservatory' temperature sensor:

http://192.168.0.100/api/getTemperature?sensorname=Conservatory	{"success":true,"state":"23.70","datetime":"2021-03-30 18:16:52","bat_voltage":"4.01","bat_level":"85.00"}

The returned JSON contains 5 components, 'success' – either true or false depending on if the API call was successful or not, 'state' – the current sensor temperature value , 'datetime' the time the sensor was last updated, 'bat_voltage' – current battery voltage, 'bat_level' – current battery level.

getZoneStatus API

Used to return the current status from a zone, the response depends on whether or not the temperature sensor attached to the zone is mains or battery powered.

Example 1 - get the current status from the 'Hot Water' zone, which uses a mains powered temperature sensor:

http://192.168.0.100/api/getZoneStatus?zonename=Hot%20Water	{"success":true,"status":"0","temp":"29.5","datetime":"2021-03-30 14:39:15"}

The returned JSON contains 4 components, 'success' – either true or false depending on if the API call was successful or not, 'status' – the current ON/OFF status of the zone, 'temp' – the temperature returned by the temperature sensor attached to the zone, 'datetime' the time the temperature sensor was last updated.

Example 2 - get the current status from the 'Central Heating' zone, which uses a battery powered temperature sensor:

http://192.168.0.100/api/getZoneStatus?zonename=Central%20Heatingr	{"success":true,"status":"0","temp":"29.5","datetime":"2021-03-30 14:39:15","bat_voltage":"4.01","bat_level":"85.00"}

The returned JSON contains 6 components, 'success' – either true or false depending on if the API call was successful or not, 'status' – the current ON/OFF status of the zone, 'temp' – the temperature returned by the temperature sensor attached to the zone, 'datetime' the time the temperature sensor was last updated, 'bat_voltage' – current battery voltage, 'bat_level' – current battery level.

modeSet API

Used to return the current operating 'Mode' of the system or set a new operating 'Mode'.

Example 1 - get the current operating 'Mode':

http://192.168.0.100/api/modeSet	`{"success":true,"mode":"TIMER"}`

The returned JSON contains 2 components, 'success' – either true or false depending on if the API call was successful or not, 'mode' – the current operating 'Mode' of the system.

Example 2 - set the operating 'Mode' to 'HW':

http://192.168.0.100/api/modeSet?mode=hw	`{"success":true,"mode":"hw"}`

The returned JSON contains 2 components, 'success' – either true or false depending on if the API call was successful or not, 'mode' – the new operating 'Mode' of the system.

Note: The parameter passed to the API can be either numeric 0 – 4 (represent the mode) or a character string "off, timer, ce, hw or both" and can be either lower or upper case.