Web API 2.02

Revised: 7th May 2009

Overview

This document outlines the available services of the AlertMe Web Application Programmers Interface (API) 2.0. The API is an XML-RPC interface that is available over a secure web connection.

The purpose of the Web API is to give external applications access to the most frequently used functions of the AlertMe web application. The functionality is of three main types:

- Access to information about the system. For example, “what devices are attached to my hub?”
- Access to data associated with the user and their hubs, devices and services. For example, “what’s the temperature in my hall?” or “is my intruder alarm armed?”
- Ability to send commands to the hub. For example, “disarm the intruder alarm.”

Location

The API is available at https://api.alertme.com/webapi/v2

Sample code

Sample code is available at https://api.alertme.com/webapi/test/v2

Rate limiting

Requests to the API are rate limited. Should a request fail because of rate limiting, you will see the following HTTP error:

503 Service Temporarily Unavailable

It is therefore important to handle this error gracefully.

The extent to which requests are limited can be reviewed on a case-by-case basis. If you believe your application needs less restrictive rate limiting, please contact support@alertme.com.

Definition

All return types are strings and all arguments are strings unless specified otherwise.

Errors

Errors are returned as XML-RPC faults. For example, an error indicating that invalid arguments were specified would be returned as follows:

```
<?xml version="1.0" encoding="utf-8"?>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>412</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string>invalid_arguments,deviceId</string></value>
        </member>
      </struct>
    </value>
  </fault>
```

The following strings may be returned to indicate errors:

Returned string	**Meaning**
no_session	Generated if a call to the API is made with an invalid, expired or null session token.
invalid_user_details	Indicates an attempt to login with an incorrect username and / or password.
internal_error	Indicates an unknown internal error. You may wish to contact AlertMe for further diagnosis of the problem or alternatively retry the request later.
service_not_available_for_login_status	Returned if the user has not yet been logged in. This will only be seen where an unauthenticated session token is created via another AlertMe API (for example in situations where AlertMe initiates phone calls to end users). Most users will never see this message.
invalid_arguments	Invalid arguments were passed to the API call – for example, a required argument may be missing or an incorrectly formatted argument may have been provided (such as a non-numeric hubId).
invalid_method	The XML RPC method that was called does not exist.
no_hub	Generated if an API call is made to control or request information from a hub but the user doesn't have one installed yet.
hub_not_contactable	Issued if a call is made that requires communications with the hub but it cannot be contacted for some reason.
invalid_hub_id	Returned if a provided hub ID does not represent a hub owned by a user.

Returned string	Meaning
needs_hub_upgrade	Some features are only available with certain hub versions. If the hub's installed software version is not sufficient to execute a particular feature then this error will be returned.
device_not_present	Generated if a device is not present but an API call is issued that depends upon it.
login_failed_account_locked	Generated if more than 5 incorrect login attempts are made on an account. Once this has occurred the user will have to arrange for a password reset before they can log in again.
privileged_systems_only	Some private extensions to this API exist for the use of AlertMe partners only. This error will be returned if an attempt is made to use them from an unauthorised source or using an inappropriately generated sessionToken.

If the API is accessed via anything other than an HTTP Post (e.g. via a Get instead), a human readable HTML error will be returned.

login

Enables customers to log in to the API service and initiates a secure session for the user. A session token is returned after a successful login call. After 20 minutes of inactivity the API will log the user out and invalidate the session token.

The caller description will be noted down in log messages that may be seen by the end user. It should be a short (ideally single word) representation of the system using this API. For example “Mobile Gadget” which may appear in logs ultimately as “The IntruderAlarm was disarmed by Joe Smith via Mobile Gadget”. The IP address of the API caller will also be noted in the logs.

Arguments:
- username
- password
- callerDescription

Returns:
- sessionToken

logout

Allows you to explicitly end your session.

Arguments:
- sessionToken

Returns:
- “ok”

getUserInfo

Returns user information.

Arguments:
- sessionToken

Returns:
- a string with the format “firstname|<first name>,lastname|<last name>,username|<username>”

getAllHubs

Returns a list of all hubs associated with the logged in user.

Arguments:
- sessionToken

Returns:
- a string with the format “<hub name 1>|<hub ID 1>,<hub name 2>|<hub ID 2>,...”

setHub

Once this method has been called, subsequent API commands in this session will operate on the specified hub. If this method isn’t called at the start of an API session, it will default to the first hub that was configured for that user.

Arguments:
- sessionToken
- hubId

Returns:
- “ok”

getHubStatus

Requests status information for the user’s selected hub. The elements of status information, with possible values in brackets, are:

- IsAvailable (‘yes’ or ‘no’):indicates whether the hub is connected to AlertMe’s servers
- IsUpgrading (‘yes’ or ‘no’): indicates whether the hub is upgrading
- PowerType (‘AC’, i.e. mains power, or ‘battery’)
- ConnectionType (‘broadband’ or ‘GPRS’)
- Uptime (since the hub booted up)

Arguments:
- sessionToken
- statusElement: Can be either NULL, or a single element from the above list.

Returns one of the following:
- the status of the element specified
- if no element was specified, a list with the format “IsAvailable|<’yes’ or ‘no’>,<IsUpgrading|<’yes’ or ‘no’>, ...”

getAllBehaviours

Returns a list of all possible behaviours for the user's selected hub.

The current list of behaviours is:
- Home
- Away
- Night

Arguments:
- sessionToken

Returns:
- a list with the format "Home,Away,Night,..."

getBehaviour

Returns the current behaviour of the user's selected hub. Note that the behaviour only changes when grace periods have timed out. For instance a hub that is arming will still be considered "Home" until the grace period expires, at which point it will be changed to "Away" (or "Night").

Arguments:
- sessionToken

Returns:
- the behaviour name (which will be one of the behaviours returned by getAllBehaviours)

getEventLog

Returns entries from the event log.

If the caller supplies a start date then up to numEntries logs will be returned (provided that they occurred before the end date if one is specified). If a start date is not specified then the most recent logs (prior to the end date if one is specified) will be returned. To get just the newest logs then only specify numEntries. The string "null" must be used to indicate that a parameter is not specified.

Note that an absolute maximum of 50 log entries will be returned per call.

Arguments:
- sessionToken
- serviceName: specify a service name if you want log entries associated with that service only, or "null" if you don't want any filtering to occur.
- numEntries (Integer): the number of entries required (max 50)
- start (UNIX time): the time for the logs to start from
- end (UNIX time): the time for the logs to end
- localisedTime: (optional) if passed as "true" then timestamps will refer to the local time of the hub itself rather than UTC/GMT.

Returns either:
- a series of comma separated log entries in the format below. (Note that some logs don't have an initiating zigbee-based device, in which case the centre section of the log response is blank): "<timestamp>|[zigbeeId|<zigbeeId>|devType|<deviceType>]|<message>, ...".

- “no_data” if there are no log messages fitting the specified criteria.

Times are in UNIX time format (number of seconds since the Epoch) and refer to UTC unless localisedTime is used.

getAllServices

Returns a list of all services enabled for the user's hub. Note that services that have been disabled by the user via the web-based AlertMe configuration system will not be listed in the response. Example services are:

- IntruderAlarm
- EmergencyAlarm
- Doorbell

AlertMe will be increasing the number of services soon. In addition, externally-provided plug-in services can be added to the system, each with their own commands that can be sent via the “sendCommand” method.

Arguments:
- sessionToken

Returns:
- a list of services with the format “<serviceName1>,<serviceName2>,…”

getAllServiceStates

Returns a list of all possible states for a specified service.

Arguments:
- sessionToken
- serviceName (e.g. “IntruderAlarm”)

Returns:
- a string with the format “<state1>,<state2>,…”

getCurrentServiceState

Requests the current state for either a single named service or all available services if no service name is specified.

Arguments:
- sessionToken
- serviceName (e.g. “IntruderAlarm”) or NULL

Returns one of the following:
- the state of the service specified
- if no service was specified, a string with the format “<serviceName>|<state>, <serviceName>|<state>, ...”

sendCommand

Sends a command to a service on the hub. Currently available commands are:

- IntruderAlarm
  - arm: set the system to away (if sent multiple times then warning/grace timeouts will be skipped)
  - disarm: set the system to home
  - nightArm: set the system to night (if sent multiple times then warning/grace timeouts will be skipped)
  - serverClear: clear the warning about the alarm having gone off whilst it was armed
- EmergencyAlarm
  - serverClear: reset the emergency alarm state
- Energy
  - on (followed by SmartPlug deviceId): switches the specified SmartPlug on
  - off (followed by SmartPlug deviceId): switches the specified SmartPlug off

Arguments:
- sessionToken
- service (e.g. “IntruderAlarm”)
- command (eg “disarm”)
- deviceId (only needed for Energy command)

Returns:
- the results of performing the command, typically “ok” or “failed”.

getAllDevices

Returns a list of all devices in the user’s system.

Arguments:
- sessionToken

Returns:
- a list of devices with the format “<deviceName1>|<deviceId1>|<deviceType1>,<deviceName2>|<deviceId2>|<deviceType2>,...”

getDeviceDetails

Returns details about a particular device.

Arguments:
- sessionToken
- deviceId

Returns:
- a string with the format “Type|<device type>,Id|<device Id>,SoftwareVersion|<software version>,HardwareVersion|<hardware version>”

getAllDeviceChannels

Returns the channels available on a particular device. Channels are types of data that a particular device provides. Examples are “Temperature”, “LQI”, “BatteryLevel” or “PowerLevel” (for energy measuring devices).

Arguments:
- sessionToken
- deviceId

Returns:
- a list of channels for that device in the format “<channel1>,<channel2>,...”

getDeviceChannelLog

Returns entries from the device log. The range parameters are handled in the same fashion as for getEventLog.

Arguments:
- sessionToken
- deviceId
- channelName (e.g. “Temperature”, “LQI” or “BatteryLevel”)
- numEntries (Integer): the number of entries required
- start (UNIX time): the time for the logs to start from
- end (UNIX time): the time for the logs to end

Returns either:
- a string with the format “<date>|<log>,<date>|<log>,…”
- “no_data” if there are no log messages fitting the specified criteria.

All times must be in UTC/GMT.

getDeviceChannelValue

Returns the current values for either a single named channel on a particular device or all available channels if no channel name is specified.

Arguments:
- sessionToken
- deviceId
- channelName (e.g. “Temperature”) or NULL

Returns one of the following:
- a string representation of the value requested for a particular channel
- if no channel was specified, a string with the format “Presence|<True/False>,Tamper|<True/False>,Upgrade|<upgradestate>,BatteryLevel|<battery voltage>,LQI|<link quality indicator level (0-100%)>,Temperature|<temperature>[,PowerLevel|<instantaneous power consumption in Watts>]”