API Reference

Introduction

The up.time API provides you with an easy way to access your up.time inventory and active status information from third-party tools. The API is designed in a general RESTful API format, allowing you to access it using any programming language that can consume URL request output.

> The up.time API is provided by the up.time Controller which is a separate installation from the up.time Monitoring Station. Please follow Installing the up.time Controller to set up your Controller installation before attempting to use the API.

In addition to the reference below, the following resources are available to help you get started with the API.

API Examples

Each API example is available as an open source project on github as well as posted as an install ready plug-ins on The Grid.

- Mobile UI - github - The Grid
- Drag & Drop Map Dashboard - github - The Grid
- Incident Console - github - The Grid
- jQuery Example Library - github

Helper Classes

Helper classes that simplify access to and management of API functions are available here:

- .Net Helper Class
- PHP Helper Class
- JavaScript Helper Class

API Overview

All API commands can be accessed using this general format:

```
https://<hostname>:<port>/api/<api_version>/<end_point>/<id>/<task>
```

- `hostname`: the URI for the up.time Controller installation
- `port`: the up.time Controller listener port, typically 9997
- `api_version`: the version of the API to run commands against (see Version Control below)
- `end_point`: the type of object you want to work with in up.time (see End Points below)
- `id`: the numerical ID of the object you are interested in
- `task`: where supported, specific tasks to be executed against the provided object

Methods

Accessing the standard format URL using different HTTP methods produces different results on the target object:

- `POST`: add an object or trigger an action
- `PUT`: edit an object
- `DELETE`: delete an object
- `GET`: view the details of an object

Currently only `GET` operations are supported.

Version Control

The version of the API you wish to access is embedded directly into the URL. Bug fixes and non-breaking feature changes will be made without changing the version number. Major feature changes or breaking changes will introduce a new version number. Backwards support for previous API version is currently not defined.

Non-breaking changes include:

- adding fields or functions to input requests
- adding fields to returned data

Breaking changes include:

- changing or removing fields or functions on input requests
- changing or removing fields in returned data

- changing the URI of any existing function

Authentication

Authentication to the API is based on the basic HTTP basic authentication template: each request to the API must provide a username and password pair. To safeguard this information, all requests to the up.time API must use SSL communication.

Each user in up.time has access to the API. User-visibility and role-based permissions are applied to each API call, ensuring users are only able to access and modify the same information they would be able to access from the up.time UI.

Endpoints

Endpoints define the different types of up.time objects that you can work with using the API. The currently supported endpoints include the following:

- elements: manage up.time Elements
- monitors: manage up.time service monitors
- groups: manage Element groups in up.time

For more information, see the following sections:

- Working with Elements
- Working with Groups
- Working with Service Monitors

Returned Results

All returned results are provided in JSON format. Successfully completed requests will return an HTTP code in the 200 range.

Error Handling

If the provided command produces an error, an HTTP status in the 400 range will be returned including a specific status code and a message with further details about the error. All error messages are returned in JSON format and look like the following:

```
{
   "error": "Required field missing",
   "errorDescription": null
}
```

Here is a quick reference of supported error codes.

HTTP Code	Error	Details	Message Examples
400	Bad Request	this is a catch-all for most error returns	Bad request
400	Required field missing	a required field hasn't been provided in the request	'hostname' must be provided
400	Required field has null or missing value	a required field doesn't have any input value	'hostname' must have a value
400	Integer field has non-integer value	an integer value was expected for a field	'max_rechecks' must be an Integer value
400	String field has unacceptable value	a string field has an unacceptable input value	'email' must be in standard email address format
400	Boolean field has non-boolean value	a boolean field has an unacceptable input value	'monitored' must be true or false
400	Unrecognized field	API does not recognize a field that was provided	'nmae' is not a recognized field
400	Naming Conflict	the current action will cause a duplicate object	<object> <name> already exists
400	Dependency Conflict	the current action cannot be performed due to a dependency conflict	Cannot <delete> <object> <name>, the group must be empty before deletion
400	License Violation	the current action will cause an invalid license scenario	Adding this <object> will violate your current license
401	Unauthorized	this is a catch-all for any authentication related error	The current user is not authorized to complete this request
401	Authentication Required	authentication information must be passed with the request; this error covers session timeouts and session invalidation.	An authenticated session must be provided within your request
401	Forbidden	the user is not permitted to perform this operation	The current user does not have permission to complete this operation

403	Forbidden	this operation has been forbidden due to throttling constraints	Exceeded API operations limit, please try again later
404	Object not found	the target ID does not exist	The element id '12345' does not exist
404	End point not found	the target endpoint is not supported	The provided end point is not supported
408	Request Timed Out	the request took longer than expected and was aborted; this does not necessarily mean the request failed.	Operation timed out

In rare circumstances a `503 - Service Unavailable` error may be returned. This indicates that the API is currently under heavy load and additional requests are being temporarily ignored in order to service existing requests.

Programming Guidance

When integrating with the API, the following general programming rules should be noted:

- fields will not always be returned in the same order; reference fields by name
- fields will not always be returned; validate a field is available before using it
- array results are not guaranteed to be sorted
- date and time fields will always be returned in ISO 8601 format