Hyperglance REST API (HGAPI)

Technical Reference

Version 0.4

June 2014

realstatus
IT insight

Table of Contents

Hyperglance REST API (HGAPI) 1
Getting Started 3
Overview 3
Concepts & Terminology 3
Authentication 5
Accessing the HGAPI 6
HGAPI v1.0 7
Network API - Get Network 7
Topology API - List 9
Topology API - Create/Update 11
Topology API - Remove 14
Augmentation API - Attributes 14
Python Code Samples 16
Running the Samples 16

Getting Started

Overview

The Hyperglance API (HGAPI) is a RESTful JSON web service API to support integrations with 3rd party applications; HGAPI client applications integrate external data and processes into the Hyperglance platform.

The HGAPI is exposed by the Hyperglance Server (HGS). By default the API's root endpoint is located at: *`https://<host>/hgapi`*

Note: We strongly recommend that client applications be configurable, in case this endpoint URL is ever changed in future versions of Hyperglance, or by administrators.

Developers should be aware when parsing responses from the HGAPI that these responses may contain a superset of JSON fields compared to those documented. HGAPI clients should be prepared to simply ignore any extra undocumented fields.

Concepts & Terminology

Datasource	A datasource is a unique name assigned to an HGAPI application and so serves to identify the source of topology data in Hyperglance. An HGAPI application has a single datasource name. Typically the name is chosen based on the underlying system from which data is retrieved. You will usually want to hard-code this into your application. Examples: "AcmeApp", “OpenStack”, "John's MongoDb"
Topology	A topology is a well-formed piece of the overall Network. Every client application contributes at least one topology. Multiple topologies can be added/updated/removed by HGAPI client applications. The use of multiple topologies by a single client application is supported purely for organizational purposes. A topology is a graph structure consisting of Nodes, Endpoints and Links. Each of which can have Attributes.
Network	An aggregation of all topologies which occurs periodically within the HGS.

Nodes	A kind of entity often used to model network devices or top-level concepts. Examples: Servers, Virtual-Machines, Load-Balancers.
Endpoints	A kind of entity often used to model hardware components or second-level concepts. An endpoint always belong to a node. A node can own many endpoints. Examples: Network Interface Cards, Ports, Logical Disk Partitions.
Links	A kind of entity often used to model connectivity or relationships between endpoints. A link always spans between two endpoints. Examples: SNMP Connectivity, Logical relationships.
Attributes	Textual key/value data. All nodes, endpoints and links support attributes.
Types	Every Node, Endpoint or Link has a type. A type is a string used to indicate the kind/flavor/variety of an entity. Typically filters will use types to assign icons & colors. A type can be any value the client application wishes although the Hyperglance Client may set default icons on nodes of certain types in the absence of a filter. Examples: vm, host, switch, connection, flow
Keys	An ID provided by the client application for every entity and must be unique within the context of a single topology. Used to assign relationships between entities within a topology (e.g. Endpoints assigned to a node) and to track the lifetime of entities across updates to the topology. Keys should be 'stable', meaning the same logical entity should be given the same key across updates to the topology.
Unique IDs (UIDs)	An ID generated by the HGS for every entity and is completely unique within the entire network. The format and structure of UID strings is undefined.
Collector-plugin	Closed-source Hyperglance Server plugins developed by Real-Status that also integrate with 3rd party applications rather like an HGAPI client application.

Authentication

Developers must generate an API Key for their application to use for authenticating with the HGS. Please remember that this API Key should be kept secret & secure and treated like a password. Every HGAPI client application should be given its own, unique API Key.

API Keys can be created and managed by visiting the HGAPI Management page:
`http://<host>/hgs/hgapi`

The HGAPI uses Basic-Access-Authentication with HTTPS for secure access. HGAPI client applications should provide the required "Authorization" header encoding their unique Datasource and API Key as the username and password respectively.

Many web request frameworks have a way to generate the "Authorization" header automatically given the username and password. However the header can be constructed quite simply as follows:

1) Combine the Datasource and API Key into a single string, separated by a colon: "Datasource:APIKey"
2) Encode this string into Base64 according to the RFC-2045 specification for MIME encoding and without any line length limit.
3) Prefix the encoded string with the word Basic and a space: "Basic "
4) Supply this string as the request header named "Authorization".

For example if the Datasource is "`AcmeApp`" and the API Key is "`a15f3a32-77f3-4784-b10c-ff8b0587495b`" then the header will be:

`Authorization`	`Basic QWNtZUFwcDphMTVmM2EzMi03N2YzLTQ3ODQtYjEwYy1mZjhiMDU4NzQ5NWI=`

The authorization header must be provided in every API call made to the HGAPI.

Sample code:

- examples/rest_utils.py

Accessing the HGAPI

GET /

Issuing an HTTP GET against the root endpoint provides a way to lookup the appropriate path to the version of the HGAPI you wish to use.

Applications should always resolve paths to the versioned APIs via this call, rather than hard-coding URLs into the application.

This API is the only un-versioned call that an application will need to make.

Normal response codes: 200 OK
Error response codes: 400 Bad Request, 401 Unauthorized

Example response:

```
{
    "versions": [
        {
            "id": "1.0",
            "status": "current",
            "path": "/hgapi/v1/1.0",
            "documentation": ["http://support.real-status.com/hgapi.pdf"]
        }
    ]
}
```

JSON elements:

versions	List of Versions	HGAPI version descriptors available on the HGS.
id	String	The identifier of the version e.g. "1.0" for HGAPI v1.
status	String	Indicates support level and availability. "current" - The API is the latest version supported the HGS "deprecated" - The API is functional but older version and may one day be removed. Other values may be used to indicate more unique scenarios.
path	String	The URL or relative path (relative paths are relative to the host) for the versioned API.
documentation	List of Strings	References human-readable documentation resources.

Sample code:

- examples/locate_hgapi_version.py

HGAPI v1.0

All paths are specified relative to the versioned API endpoint that is retrieved via the root API endpoint (see 'Accessing the HGAPI').

Network API - Get Network

GET /network

Returns topologies from all datasources that have been aggregated into a coherent network that the Hyperglance Client could display. Topologies are grouped by datasource and can be filtered to a given datasource if the optional query parameter is provided.

Note: Due to the aggregation process, the topologies returned by this call may have been altered since they were originally contributed.

Note: The aggregation process is an automatic, periodic process so this call may not immediately reflect changes to a topology.

Normal response codes: 200 OK
Error response codes: 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found

Query parameters:

datasource (optional)	The unique name identifier of a data-source to constrain the response to. Only the part of the network contributed by the provided data-source will be returned.

Example response:

```
{
    "updatedAt": 123456789,
    "network": [
        {
            "datasource": "AcmeApp",
            "topologies": [
                {
```

```
                    "name": "example topology"
                    "nodes": [
                        {
                            "UID": "902c-8ce241a85bbb",
                            "key": "node1",
                            "type": "vm",
                            "attributes": [
                                { "name": "Name", "value":"new-vm-test" }
                            ]
                        },
                        {
                            "UID": "9421-8bb2a8139669",
                            "key": "node2",
                            "type": "host",
                            "attributes": [
                                { "name": "Name", "value":"lab host1" }
                            ]
                        }
                    ],
                    "endpoints": [
                        {
                            "UID": "8c73-e7f06f359a1f",
                            "key": "ep1",
                            "type": "virtual port",
                            "attributes": [],
                            "nodeUID": "902c-8ce241a85bbb"
                        },
                        {
                            "UID": "988a-60aa3b19f13a",
                            "key": "ep2",
                            "type": "physical port",
                            "attributes": [],
                            "nodeUID": "9421-8bb2a8139669"
                        }
                    ],
                    "links": [
                        {
                            "UID": "becd-5c6d4c62b09c",
                            "key": "host-to-vm link",
                            "type": "connection",
                            "attributes": [],
                            "endpointAUID": "8c73-e7f06f359a1f",
                            "endpointBUID": "988a-60aa3b19f13a"
                        }
                    ]
                }
            ]
        }
    ]
}
```


JSON elements:

updatedAt	Integer	Time the Network was last updated. Unit: Seconds since Unix Epoch.
network	List of Datasource-Topologies	Topologies grouped by their data-source identifier.
datasource	String	The data-source identifier.

topologies	List of Topologies	All topologies contributed by a data-source. Absent from these topologies are any non-creatable entities.
name	String	Name of topology.
nodes	List of Nodes	Nodes in a topology.
endpoints	List of Endpoints	Endpoints in a topology.
links	List of Links	Links in a topology.
UID	String	The UID of the Node, Endpoint or Link.
key	String	The Key of the Node, Endpoint or Link.
type	String	The Type of the Node, Endpoint or Link.
attributes	List of Attributes	Attributes of the Node, Endpoint or Link.
name	String	The name of an Attribute.
value	String	The value of an Attribute.
nodeUID	String	The UID of the Node that an Endpoint attaches to.
endpointAUID	String	The UID of an Endpoint that a Link attaches to.
endpointBUID	String	The UID of an Endpoint that a Link attaches to.

Sample code:

- examples/get_network.py
- examples/augment_attributes2.py

Topology API - List

GET /topology

Lists all topologies as contributed by the HGAPI client application.

Normal response codes: 200 OK
Error response codes: 400 Bad Request, 401 Unauthorized, 403 Forbidden

Example response:

```
{
    "topologies": [
        {
            "name": "example topology"
            "nodes": [
                {
                    "UID": "902c-8ce241a85bbb",
                    "key": "node1",
                    "type": "vm",
                    "attributes": [
                        { "name": "Name", "value":"new-vm-test" }
```

```
                    ]
                },
                {
                    "UID": "9421-8bb2a8139669",
                    "key": "node2",
                    "type": "host",
                    "attributes": [
                        { "name": "Name", "value":"lab host1" }
                    ]
                }
            ],
            "endpoints": [
                {
                    "UID": "8c73-e7f06f359a1f",
                    "key": "ep1",
                    "type": "virtual port",
                    "attributes": [],
                    "nodeKey": "node1",
                    "nodeUID": "902c-8ce241a85bbb"
                },
                {
                    "UID": "988a-60aa3b19f13a",
                    "key": "ep2",
                    "type": "physical port",
                    "attributes": [],
                    "nodeKey": "node2",
                    "nodeUID": "9421-8bb2a8139669"
                }
            ],
            "links": [
                {
                    "UID": "becd-5c6d4c62b09c",
                    "key": "host-to-vm link",
                    "type": "connection",
                    "attributes": [],
                    "endpointAKey": "ep1",
                    "endpointAUID": "8c73-e7f06f359a1f",
                    "endpointBKey": "ep2",
                    "endpointBUID": "988a-60aa3b19f13a"
                }
            ]
        }
    ]
}
```


JSON elements:

topologies	List of Topologies	All topologies contributed by this datasource.
name	String	Name of topology.
nodes	List of Nodes	Nodes in a topology.
endpoints	List of Endpoints	Endpoints in a topology.
links	List of Links	Links in a topology.
UID (optional)	String	The UID of the Node, Endpoint or Link. Will be null if the entity is not creatable: E.g. An Endpoint is not creatable if its nodeKey or nodeUID are invalid.

key	String	The Key of the Node, Endpoint or Link.
type	String	The Type of the Node, Endpoint or Link.
attributes	List of Attributes	Attributes of the Node, Endpoint or Link.
name	String	The name of an Attribute.
value	String	The value of an Attribute.
nodeKey (optional)	String	The Key of the Node that an Endpoint attaches to. Will be null if one was not specified.
nodeUID	String	The UID of the Node that an Endpoint attaches to.
endpointAKey (optional)	String	The Key of an Endpoint that a Link attaches to. Will be null if one was not specified.
endpointAUID	String	The UID of an Endpoint that a Link attaches to.
endpointBKey (optional)	String	The Key of an Endpoint that a Link attaches to. Will be null if one was not specified.
endpointBUID	String	The UID of an Endpoint that a Link attaches to.

Sample code:

- examples/list_topologies.py

Topology API - Create/Update

PUT /topology

Creates or updates (if already created by the HGAPI client application) a single topology.

Normal response codes: 200 OK, 202 Accepted
Error response codes: 400 Bad Request, 401 Unauthorized, 403 Forbidden

Example request:

```
{
    "name": "example topology"
    "nodes": [
        {
            "key": "node1",
            "type": "vm",
            "attributes": [
                { "name": "Name", "value":"new-vm-test" }
            ]
        },
        {
            "key": "node2",
```

```
            "type": "host",
            "attributes": [
                { "name": "Name", "value":"lab host1" }
            ]
        }
    ],
    "endpoints": [
        {
            "key": "ep1",
            "type": "virtual port",
            "nodeKey": "node1"
        },
        {
            "key": "ep2",
            "type": "physical port",
            "nodeKey": "node2"
        }
    ],
    "links": [
        {
            "key": "host-to-vm link",
            "type": "connection",
            "endpointAKey": "ep1",
            "endpointBKey": "ep2"
        }
    ]
}
```

JSON elements:

name	String	The unique name of the topology being created/updated. Must be unique within the scope of the datasource.
nodes (optional)	List of Nodes	Nodes to be created/updated. Any nodes created prior for this topology and are not present in this list will be removed.
endpoints (optional)	List of Endpoints	Endpoints to be created/updated. Any endpoints created prior for this topology and are not present in this list will be removed.
links (optional)	List of Links	Links to be created/updated. Any links created prior for this topology and are not present in this list will be removed.
key	String	An identifier that must be unique within this topology. Valid on Nodes, Endpoints and Links.
type	String	A type indicator. May take any value. Valid on Nodes, Endpoints and Links.
attributes (optional)	List of Attributes	Attributes to be created/updated Valid on Nodes, Endpoints and Links.
name	String	The name of an attribute to be created/updated.
value	String	The value of an attribute to be created/updated.
nodeKey (optional)	String	The Key of the node to attach to. Only valid on Endpoints. One of **nodeKey** or **nodeUID** must be provided.
nodeUID (optional)	String	The UID of the node to attach to.

		Only valid on Endpoints. One of **nodeKey** or **nodeUID** must be provided.
endpointAKey (optional)	String	The Key of an endpoint to attach one end of a link to. Only valid on Links. One of **endpointAKey** or **endpointAUID** must be provided.
endpointAUID (optional)	String	The UID of an endpoint to attach one end of a link to. One valid on Links. One of **endpointAKey** or **endpointAUID** must be provided.
endpointBKey (optional)	String	The Key of an endpoint to attach one end of a link to. Only valid on Links. One of **endpointBKey** or **endpointBIUD** must be provided.
endpointBUID (optional)	String	The UID of an endpoint to attach one end of a link to. Only valid on Links. One of **endpointBKey** or **endpointBUID** must be provided.

Example response:

```
{
    "nodes": [
        { "key":"node1" "UID": "902c-8ce241a85bbb" },
        { "key":"node2" "UID": "9421-8bb2a8139669" }
    ],
    "endpoints": [
        { "key": "ep1", "UID": "8c73-e7f06f359a1f" },
        { "key": "ep2", "UID": "988a-60aa3b19f13a" }
    ],
    "links": [
        {"key": "host-to-vm link", "UID": "becd-5c6d4c62b09c" }
    ]
}
```

JSON elements:

nodes	Nodes that were created or updated in this topology.
endpoints	Endpoints that were created or updated in this topology.
links	Links that were created or updated in this topology.
key	The Key of the Node, Endpoint or Link that was created or updated.
UID	The generated unique UID of the Node, Endpoint or Link that was created or updated. Note: UIDs will not change once an entity is created; the UIDs of updated entities are the same as they were when those entities were first created.

Sample code:

- examples/create_topology.py

Topology API - Remove

DELETE /topology

Removes all topologies contributed by the HGAPI client application, or removes a single topology if called with a query parameter specifying an individual topology name.

Normal response codes: 202 Accepted, 204 No Content
Error response codes: 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found

Query parameters:

name (optional)	The name of a topology to remove. (Must be a topology that was contributed by the HGAPI client application).

Sample code:

- examples/delete_topology.py

Augmentation API - Attributes

PUT /augment/attributes

Specifies the set of attribute augmentations to apply to any pre-existing nodes, endpoints or links. Replacing all attribute augmentations provided in any prior invocation of this call made by the HGAPI client application.

Normal response codes: 200 OK, 202 Accepted
Error response codes: 400 Bad Request, 401 Unauthorized, 403 Forbidden

Example request:

```
{
    "attributes": [
        {
            "UID": "acda-dee6d506a1db",
            "attributes": [
                { "name": "Contact E-Mail", "value": "jon@example.com" }
            ]
        }
```

```
    ]
}
```

JSON elements:

attributes	List of Attribute-Augmentations	Attribute-Augmentations to apply. Any Attribute-Augmentations created prior by this data-source will be replaced.
UID	String	The UID of Node, Endpoint or Link receiving the augmentation.
attributes	List of Attributes	Attributes to add to the Node, Endpoint or Link. Any Attribute-Augmentations created prior for the entity by this data-source will be replaced.
name	String	The name of an attribute.
value	String	The value of an attribute.

Sample code:

- examples/augment_attributes1.py
- examples/augment_attributes2.py

Python Code Samples

A selection of Python code samples that demonstrate the use of the API are located in the SDK*/examples* subdirectory.

These samples were written against **Python 3.4**, but may work in newer versions too.

Running the Samples

1. Open a browser and visit: *https://<host>/hgs/hgapi*
2. Register a new App by entering a suitable Datasource name (for example call it: AcmeApp) and giving it a helpful label/description.
3. Note the API key that is generated. Keep this safe and treat it like a password!
4. Open *SDK/examples/config.py* in a text-editor.
5. Set the `DATASOURCE` and `API_KEY` values to match those from steps 2 and 3 above.
6. Set the `HGS_SERVER` value to the URL of your Hyperglance Server (including the port if necessary).
7. Ensure that the Python interpreter is on you PATH. On Windows Python 3.4 is installed under: *C:\Python34*
8. To run a Python script, launch a command-line shell and type: `python name_of_script.py`