The Open-AudIT API

Introduction

Open-AudIT has a JSON Restful API to be used both in the web interface and via JSON requests.

Open-AudIT's API

Open-AudIT's API is base upon http://jsonapi.org with the intention of providing simple and intuitive access in a manner familiar to developers.

In addition to this API, the web interface will use the same request format and supply some additional actions (eg: HTML forms for creating items).

Access Model

The API uses a cookie. You can request a cookie by sending a POST to the URL below, containing the username and password attributes and values:

```
http://{server}/open-audit/index.php/logon
```

POSTing data

To create a resource, you should POST the required data.

When POSTing data, you must include an access token. An access token is generated with every request type, so make a GET (for example) and Accept: application/json, parse the response for metaaccess_token, and include that with your request. This should be placed in the field data[access_token], IE, the top level.

The format of your data should be in the form:

`data[attributes][ATTRIBUTE_NAME]`

You should substitute the required column (eg, org_id) for ATTRIBUTE_NAME.

In the case where we store several fields (usually in JSON format) inside a BIGTEXT MySQL field (eg: credentials.credentials - the credentials column in the credentials table), you should use the format:

`data[attributes][credentials][credentials][username]`

Som examples are at the bottom of this page.

All endpoints also have a minimum list of required fields. These are:

applications: name,org_id

attributes: name,org_id,type,resource,value

baselines: name,org_id

buildings: name,org_id,location_id

clouds: name,org_id,type,credentials,options

clusters: name,org_id

collectors: name,org_id,type,host,community,username,password

connections: name,org_id

credentials: name,org_id,type,credentials

dashboards: name,org_id,options,sidebar

devices: name,org_id

discoveries: name,org_id,type

discovery_scan_options: name,org_id,ping,service_version,filtered,open|filtered,timing,nmap_tcp_ports,
nmap_udp_ports

fields: name,org_id,type

files: name,org_id,path

groups: name,org_id,sql

integrations: name,org_id,attributes,fields

ldap_servers: name,org_id,lang,host,port,secure,domain,type,version,use_auth,use_roles,refresh

licenses: name,org_id,org_descendants,purchase_count,match_string

locations: name,org_id

networks: name,org_id,network

orgs: name,parent_id

queries: name,org_id,sql,menu_category,menu_display

racks: name,org_id,ru_height

rack_devices: rack_id,device_id,position,height

roles: name,permissions

rules: name,org_id

scripts: name,org_id,options,based_on

summaries: name,org_id,table,column,menu_category

tasks: name,org_id,type,sub_resource_id,uuid,enabled,minute,hour,day_of_month,month,day_of_week

users: name,org_id,lang,roles,orgs

widgets: name,org_id,type

An example JSON POST body is below. This should be attached to the "data" form item.

```
{
  "access_token": "bbc0c85653fdc4b83d108cba7641bfcbbc77586dfb8f32d08973770a90fe",
  "type": "discoveries",
  "attributes": {
    "name": "My Test Discovery",
    "type": "subnet",
        "subnet": "192.169.1.150"
    "org_id": 1,
    "scan_options": {<removed for brevity>},
    "match_options": {<removed for brevity>},
    }
  }
}
```

The Endpoints

At present, we have endpoints for nearly every collection. They are listed here - Collections.

Options

Format

Using the format option is useful when using a web browser but you wish to see the result in JSON format. Adding format=json achieves this. If you only want the actual data in JSON, format=json_data will do the trick. Normally a web browser will set its accept header to HTML, so in that case, we return the rendered page. Using an API to retrieve JSON you should set the accept header to "json/application". You can override this by providing the format option in the URL.

We tend to use the Google Chrome extension called Postman for testing actual restful queries. You might like to install and test with that. *http://www.getpostman.com*.

Action

NOTE - Removed from 5.0.0.

When using the API the default action is determined according to the format and URL. You can override this by providing the 'action' option in the URL. An example of this is when creating a new item. You would normally use POST to /item but in the case of a web user, you need a web form to be able to fill out the item details. In that case, there is no facility for this in a typical JSON Restful API. We work around this by providing action=create in a GET request for the URL. IE - http://{server}/omk/open-audit/networks?action=create. The default action if nothing matches below is to return a collection of items.

API Routes

Request Method	ID	Action	Resulting Function	Permission Required	URL Example	Notes
POST	n		create	{collection}::create	/{collection}	Insert a new {collection} entry.
GET	y		read	{collection}::read	/{collection}/{id}	Returns a {collection} details.
PATCH	y		update	{collection}::update	/{collection}/{id}	Update an attribute of a {collection} entry.
DELETE	y		delete	{collection}::delete	/{collection}/{id}	Delete a {collection} entry.
GET	n		collection	{collection}::read	/{collection}	Returns a list of {collection}.

Web Application Routes

Request Method	ID	Action	Resulting Function	Permission Required	URL Example	Notes
GET	n	create	create_form	{collection}::create	/{collection}/create	Displays a standard web form for submission to POST /{collection}.
GET	n	import	import_form	{collection}::create	/{collection}/import	Displays a standard web form for submission to POST /{collection}/import.

POST	n	import	import	{collection}::create	/{collection}/import	Import multiple {collection} using a CSV.
GET	y	execute	execute	(collection)::see below	/{collection}/{id} /execute	Some collections can be executed - queries, etc - see below.

Execute permissions required per endpoint

Endpoint	Permission
baselines	read
clouds	read
dashboards	read
database	update
discovery	update
groups	read
queries	read
summaries	read
tasks	read

Sort

To sort by a database column, use "sort={attribute}". To reverse sort, insert a minus, thus "sort=-{attribute}".

```
sort=[-]{attribute}
```

Current

NOTE - Removed from 5.0.0. Please use components.current=n or components.current=IN('y','n') instead (if required).

By default, only attributes with "current=y" are retrieved. To override this, set current as below.

```
current={y|n|all}
```

GroupBy

NOTE - Removed from 5.0.0.

```
groupby={attribute}
```

Limit

When requesting JSON, by default no limit is set.

When requesting screen display, the limit is set to 1000 by default.

```
limit={int}
```

Offset

The offset is the count of devices you wish to return data from.

```
offset={int}
```

Properties

The requested properties should be in a comma-separated list. Properties should be fully qualified - ie, system.hostname (not just hostname).

NOTE - From 5.0.0 onwards, the system table has been replace by the devices table - so devices.name, not system.name.

```
properties=system.id,system.name,system.status
```

You can also specify properties using the below format.

```
properties=["system.id","system.name","system.status"]
```

Filter

To filter by a property value, use the property name. Operators that should precede the value are !=, >, >=, <, <=, 'like' and '!like'. If no operator is specified, the default is =. Properties should be fully qualified - ie, system.hostname (not just hostname).

```
{attribute}=[operator]{value}
```

End Points

All endpoints URLs for prior to v5 are of the format http://{server}/omk/open-audit/{endpoint}

NOTE - From 5.0.0 all endpoint URLs are of the form - http://{server}/open-audit/index.php/{endpoint}

Devices

NOTE - From 5.0.0 the sub_resource item has been replaced by the components endpoint.

Type	Endpoint v4	v5		
GET	`/system`	`/devices`	Return a collection of devices with the default set of columns from the system table (system.system_id, system.icon, system.man_type, system.hostname, system.domain, system.man_ip_address, system.man_description, system.man_os_family, system.man_status)	
GET	`/system/{id}`	`/devices/{id}`	Return an individual devices details.	
GET	`/system?sub_resource={sub_resource name}`	`/components?components.type={sub_resource name}`	To return all items in a sub_resource for a collection of devices. If you wanted all software you would use http://{server}/open-audit/index.php/devices?sub_resource=software	
GET	`/system/{id}?sub_resource={sub_resource name}`	`/components?components.type={sub_resource name}&components.device_id={id}`	To return all items in a sub_resource for a specific device.	
GET	`/system/{id}?sub_resource={sub_resource name}&sub_resource_id={sub_resource id}`	`/components/{sub_resource id}?components.type={sub_resource name}`	To return a specific sub_resource item.	

Device sub_resource names / component types

NAME	NAME	NAME
audit_log	netstat	service
bios	network	share
change_log	optical	software
credential	pagefile	software_key
disk	partition	sound
dns	print_queue	task
edit_log	processor	user
ip	radio	user_group
log	route	variable
memory	san	video
module	scsi	vm
monitor	server	windows
motherboard	server_item	

Examples

NOTE - Where there are two examples, the second is for 5.0.0 a newer versions.

NOTE #3 - You should substitute items in the URL enclosed in {} brackets with the relevant items for your environment.

Retrieve all devices with the standard columns:

```
GET http://{server}/omk/open-audit/devices
GET http://{server}/open-audit/index.php/devices
```

Retrieve all devices running Windows.

```
GET http://{server}/omk/open-audit/devices?system.os_group=Windows
GET http://{server}/open-audit/index.php/devices?devices.os_group=Windows
```

Retrieve the first 10 devices running Windows ordered by hostname

```
GET http://{server}/omk/open-audit/devices?system.os_group=Windows&limit=10&sort=system.hostname
GET http://{server}/open-audit/index.php/devices?devices.os_group=Windows&limit=10&sort=devices.hostname
```

Retrieve the properties id, ip, hostname, domain, type from all devices

```
GET http://{server}/omk/open-audit/devices?properties=system.id,system.ip,system.hostname,system.domain,system.
type
GET http://{server}/open-audit/index.php/devices?properties=devices.id,devices.ip,devices.hostname,devices.
domain,devices.type
```

Retrieve all details about the device with id 88.

```
GET http://{server}/omk/open-audit/devices/88?include=all
GET http://{server}/open-audit/index.php/devices/88
```

Retrieve a list of devices in the 192.168.1.0/24 subnet

```
GET http://{server}/omk/open-audit/devices?sub_resource=ip&ip.network=192.168.1.0/24&properties=system.id,
system.hostname,system.domain,ip.ip
GET http://{server}/open-audit/index.php/devices?ip.network=192.168.1.0/24&properties=devices.id,devices.
hostname,devices.domain,ip.ip
```

Retrieve a list of devices with OS Name like Windows 2008

```
GET http://{server}/omk/open-audit/devices?system.os_name=likeWindows 2008
GET http://{server}/open-audit/index.php/devices?devices.os_name=likeWindows 2008
```

CURL Examples

Logging in

```
curl --cookie-jar cookies.txt --form password=password --form username=admin http://{server}/open-audit/index.
php/logon
```

Creating Credentials

```
curl -X POST -b cookies.txt http://{server}/open-audit/index.php/credentials -d "data[attributes][name]
=test_creds&data[attributes][org_id]=1&data[attributes][type]=ssh&data[attributes][credentials][username]
=my_new_user&data[attributes][credentials][password]=my_new_password"
```

Retrieving a List of Credentials

```
curl -X GET -b cookies.txt http://{server}/open-audit/index.php/credentials
```

Update attributes

NOTE - The curly brackets in the data filed should be used as-is (not replaced as per other examples above).

```
curl -X PATCH -b cookies.txt -d 'data={"data":{"id":"3","type":"devices","attributes":{"description":"Test
Description"}}}' http://{server}/open-audit/index.php/devices/3
```