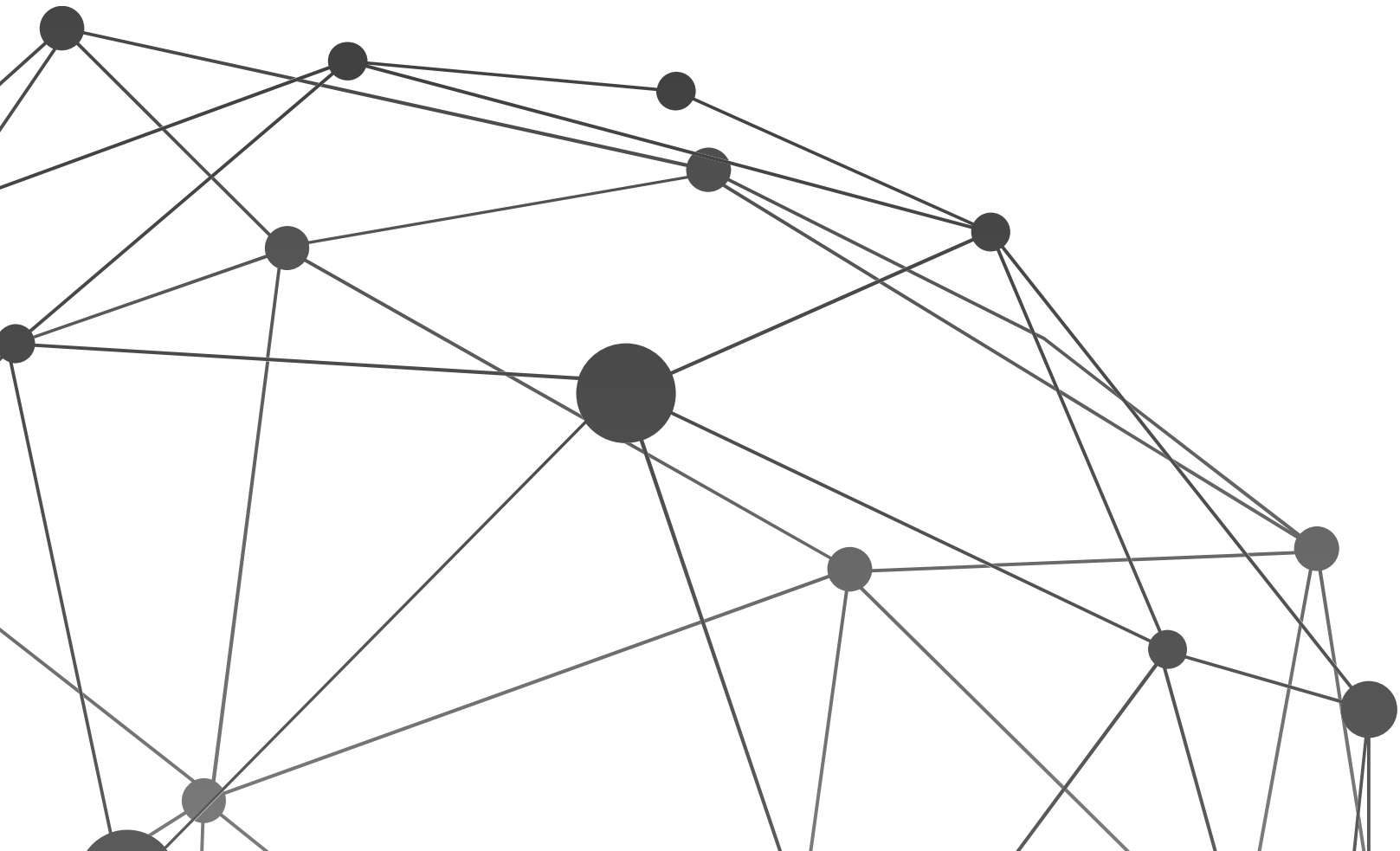
HEAT software

EMSS REST API v1.0.1

Getting Started

February 14, 2017



Notices

Copyright Information

Lumension Security Inc.,
8660 East Hartford Drive, Suite 300
Scottsdale, AZ 85255
Phone: +1 888.725.7828
Fax: +1 480.970.6323
E-mail: info@lumension.com

Trademark Information

Lumension®, *Lumension®* Endpoint Management and Security Suite, *Lumension®* Endpoint Management Platform, *Lumension®* Patch and Remediation, *Lumension®* Enterprise Reporting, *Lumension®* Security Configuration Management, *Lumension®* Content Wizard, *Lumension®* AntiVirus, *Lumension®* Wake on LAN, *Lumension®* Power Management, *Lumension®* Application Control, *Lumension®* Device Control, *Lumension®* Endpoint Security, *Lumension®* Intelligent Whitelisting, PatchLink®, PatchLink®Update™, their associated logos, and all other Lumension trademarks and trade names used here are the property of Lumension Security, Inc. or its affiliates in the U.S. and other countries.

RSA Secured® is a registered trademark of RSA Security Inc.

Apache is a trademark of the Apache Software Foundation.

In addition, any other companies' names, trade names, trademarks, and products mentioned in this document may be either registered trademarks or trademarks of their respective owners.

Feedback

Your feedback lets us know if we are meeting your documentation needs. E-mail the HEAT Software Technical Publications department at techpubs@heatsoftware.com to tell us what you like best, what you like least, and to report any inaccuracies.

Contents

Contents 4
Introduction 5
Requirements 5
Accessing the interactive REST API documentation 5
Useful tools 7
Request URLs 7
Request/Response Format 7
HTTP Methods 7
Standard Response Codes 8
Use Cases 8
Endpoints and Groups Walkthrough 13
Troubleshooting 24

Introduction

The EMSS REST API provides a simple RESTful interface with lightweight JSON-formatted responses to read and write data from EMSS. This document provides information to get you started on integrating with the EMSS REST API.

Requirements

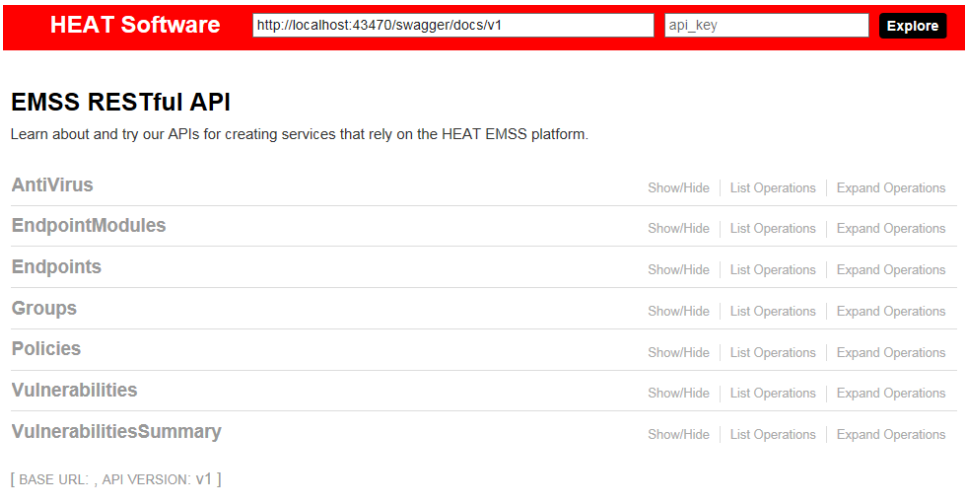
- Download and install the installer from the Support Site. You must run the installer as a user with Administrator rights.
- HEAT EMSS 8.5 server installed and running.
- At least one endpoint installed and communicating with the server.
- Access the API over HTTP or HTTPS, but HTTPS is recommended where possible.

Accessing the interactive REST API documentation

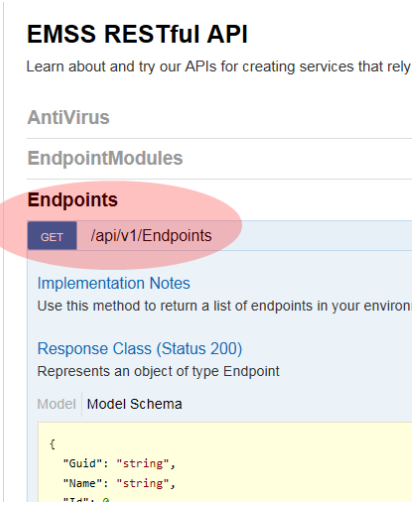
The fastest way to learn about the EMSS REST API's calls and parameters is to use the interactive Swagger™ based documentation included with the install. Within a Web browser you can specify inputs to an operation, call that operation, and inspect the results of calling that operation.

1. Open a browser and enter one of the following URLs:
    - Without SSL: **`http://<host>:<port>/docs/index`**
    - With SSL: **`https://<host>:<port>/docs/index`**

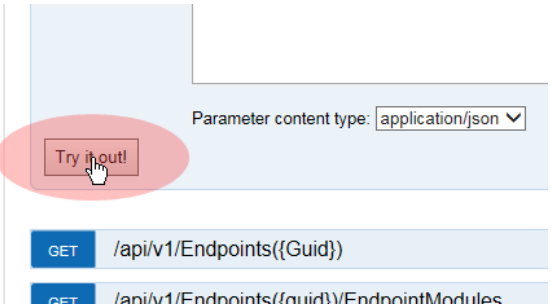
The HEAT REST API Swagger documentation appears:



2. Expand **Endpoints > GET /api/v1/Endpoints**.



3. Click **Try it out!**



Useful tools

Besides the Swagger™ docs, you can test API requests and responses using Postman, cURL, and Fiddler.

Request URLs

Each resource or resource collection in a RESTful API is identified by a unique URL. For example:

```
http://<localhost>:<port>/api/v1/Endpoints
```

v1 is the version specifier and must be present. When significant changes are made to the API (changes that would break compatibility with existing applications) this value will change.

Any web programming language (Ruby, PHP, Perl, Python, Java, Objective C, C#) can make and receive HTTP networking calls. Consult the documentation for your language of choice.

Request/Response Format

The response format used (including for error responses) is JSON, a lightweight serialization language that is compatible with many different languages.

HTTP Methods

Standard HTTP methods are used to denote actions against a resource:

GET
: Reads a resource and returns HTTP 200 on success.

POST
: Creates a new resource and returns HTTP 201 on success.

PUT
: Updates a resource and returns HTTP 200 on success.

DELETE
: Deletes a resource and returns HTTP 200 on success.

Standard Response Codes

Conventional HTTP response codes are used to indicate the success or failure of an API request.

- 2xx range codes indicate success,
- 4xx range codes indicate an error that resulted from the caller provided information (e.g. a required parameter was missing, a charge failed, etc.),
- 5xx range codes indicate an error with the server.

Errors are returned using standard HTTP error code syntax. Any additional info is included in the body of the return call, JSON-formatted.

Use Cases

These use cases describe business scenarios that you can address using the EMSS REST API.

View AntiVirus event alerts

You need to query information about the event alerts generated by virus and malware scans in your environment.

Solutions

- GET /api/v1/AvAlerts to return a list of all alerts generated in your environment.

  Request URL: `http://<host>:<port>/api/v1/AvAlerts`
- GET /api/v1/Endpoints({guid})/AvAlerts to return the alerts generated by a specific endpoint.

  Request URL: `http://<host>:<port>/api/v1/Endpoints(<Guid>)/AvAlerts`

Check the status of the AntiVirus module

You need to confirm that endpoints are protected by the AntiVirus module and their definitions are up-to-date.

Solutions

- GET /api/v1/Modules to return a list of modules installed on endpoints.

  Request URL: `http://<host>:<port>/api/v1/Modules`

- GET /api/v1/Endpoints({guid})/EndpointModules to return a list of modules installed on a specific endpoint.

  Request URL: `http://<host>:<port>/api/v1/Endpoints(<Guid>)/EndpointModules`

- GET /api/v1/Endpoints({guid})/AvDefinition to return information about the AntiVirus definitions file installed on a specific endpoint.

  Request URL: `http://<host>:<port>/api/v1/Endpoints(<Guid>)/AvDefinition`

View information about AntiVirus policies

You need information about the AntiVirus policies in your environment, like when it was created and the number of endpoints, groups and entities assigned to them.

Solutions

- GET /api/v1/Policies to get information about the AntiVirus policies in your environment.

  Request URL: `http://<host>:<port>:43470/api/v1/Policies`

- GET /api/v1/Policies{'{PolicyType}'} to return a list of AntiVirus policies of a specific type.

  Request URL: `http://<host>:<port>/api/v1/Policies(<PolityType>)`

- GET /api/v1/Endpoints({guid})/AvDefinition to return information about the AntiVirus definitions file installed on a specific endpoint.

  Request URL: `http://<host>:<port>/api/v1/Endpoints(<Guid>)/AvDefinition`

Manage groups and the endpoints within them

You need to create, delete and get information about groups, as well as add and remove endpoints from them.

Solutions

- GET /api/v1/Groups to return a list of all the groups in your environment.

  Request URL: `http://<host>:<port>/api/v1/Groups`
- GET /api/v1/Groups({Id}) to return information about a specific group.

  Request URL: `http://<host>:<port>/api/v1/Groups(<Id>)`
- GET /api/v1/Groups({id})/Endpoints to return information about the endpoints in a specific group.

  Request URL: `http://<host>:<port>/api/v1/Groups(<Id>)/Endpoints`
- DELETE /api/v1/Groups({id})/Endpoints({endpointGuid}) to delete specific endpoints in a specific group.
- DELETE /api/v1/Groups to delete an existing group.
- POST /api/v1/Groups to create a new group.
- POST /api/v1/Groups({id})/Endpoints({endpointGuid}) to add specific endpoints to a specific group.

See the Walkthrough for information on how to configure a POST request.

Identify endpoints requiring reboot

You need to know which endpoints require a reboot so your client management system can schedule the reboot to occur at a convenient time.

Solutions

- GET /api/v1/Modules with an OData filter to get a list of modules with a `IsPendingReboot` parameter value of `True`.

Request URL:
`http://<host>:<port>/api/v1/Modules?$filter=IsPendingReboot eq true`

- GET /api/v1/Endpoints({guid})/EndpointModules to get a list of modules installed on a specific endpoint and information about them. Check the `IsPendingReboot` parameter status for AntiVirus.

  Request URL: `http://<host>:<port>/api/v1/Endpoints(<Guid>)/EndpointModules`

System Synchronization

You need to run a daily query of registered endpoints to compare with a master system for inventory purposes.

Solutions

- GET /api/v1/Endpoints to get a list of endpoints in your environment.

  Request URL: `http://<host>:<port>/api/v1/Endpoints`

Check the vulnerability status of endpoints

You need to verify that endpoints are patched for all critical vulnerabilities before allowing them access to the network.

Solutions

- GET /api/v1/Endpoints({guid})/Vulnerability to return vulnerabilities for a specific endpoint.

  Request URL: `http://<host>:<port>/api/v1/Endpoints(<Guid>)/Vulnerability`

- GET /api/v1/VulnerabilitiesSummary to return a summary of the vulnerabilities patched/not patched in your environment.

  Request URL: `http://<host>:<port>/api/v1/VulnerabilitiesSummary`

- GET /api/v1/VulnerabilitiesSummary({EndpointGuid}) to return a summary of the vulnerabilities patched/not patched on a specific endpoint.

  Request URL: http://<host>:<port>/api/v1/VulnerabilitiesSummary(<EndpointGuid>)

Report on Discover Applicable Updates (DAU) scans

You need information on DAU scans to ensure they are running and you're getting accurate and up to date endpoint vulnerability statuses.

Solution

- GET /api/v1/Endpoints({guid})/Vulnerability to return information about a DAU scan on a specific endpoint.

  Request URL: `http://<host>:<port>/api/v1/Endpoints(<Guid>)/Vulnerability`

Endpoints and Groups Walkthrough

Let's dive into the REST API and see how you can use it to manage endpoints and groups in your environment.

To start this walk-through, you must have:

- EMSS installed and at least one endpoint reporting to it.
- REST API host application installed and configured correctly.
- A Web browser open.
- Swagger™ Interactive docs open.
- Postman App open.

The root URL for API calls is `http://<localhost>:<port>/api/v1/` . Your root URL will depend on the port you chose at install time and whether you are accessing the API remotely. In this walkthrough we'll use localhost and port 43470.

When accessing the REST API remotely ensure you have access to the port chosen at install time (default 43470). This may require allowing this port through the firewall of the machine the REST API is installed on and any other network firewalls or proxies.

Exercise 1: Get the entity collection of installed endpoints

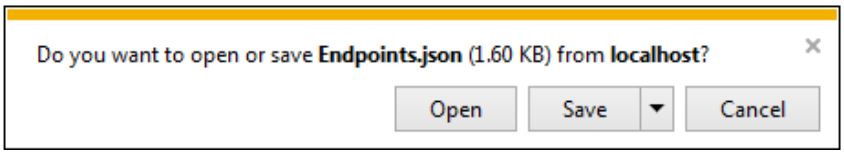
First let's test you have everything setup correctly by requesting a list of the endpoints in your environment. Enter this call into your browser:

`http://localhost:43470/api/v1/Endpoints`

The result is a JSON document containing information about the Endpoints.

```
{
  "@odata.context":"http://localhost:43470/api/v1/$metadata#Endpoints","value":[
    {
      "Guid":"8a6f1210-0b15-4da1-8ef9-d3b155d31d58","Name":"DP-M-
LATEST2012","Id":3,"Suffix":"","IpAddress":"[redacted]","MacAddress":"[redacted]
:1D:46","Version":"8.5.0.136","Enabled":true,"Status":"Online","InstallDate":"2016
-10-06T21:30:02.067+01:00","LastContactDate":"2017-02-09T13:16:54.923Z"
    },{
      "Guid":"548cb848-8dc3-4162-b895-451b0b860770","Name":"DP-M-
W10PROX86","Id":4,"Suffix":"","IpAddress":"[redacted]","MacAddress":"[redacted]
15:28","Version":"8.5.0.178","Enabled":true,"Status":"Offline","InstallDate":"2016
-10-06T21:41:13.46+01:00","LastContactDate":"2017-02-09T09:30:48.753Z"
    },{
      "Guid":"e09b922f-447a-4803-bf92-d7789742bade","Name":"DP-M-
W7PRO64","Id":5,"Suffix":"","IpAddress":"[redacted]","MacAddress":"[redacted]3C
:28","Version":"8.3.0.139","Enabled":true,"Status":"Online","InstallDate":"2016-
10-06T21:42:42.583+01:00","LastContactDate":"2017-02-09T13:13:43.76Z"
    },{
      "Guid":"51848eaa-9073-4d75-b91f-2d50e3ef1b70","Name":"DP-M-
2016ESS","Id":7,"Suffix":"","IpAddress":"[redacted]","MacAddress":"[redacted]2C
:07","Version":"8.5.0.195","Enabled":true,"Status":"Offline","InstallDate":"2016-
10-26T18:17:50.033+01:00","LastContactDate":"2017-01-12T10:27:06.607Z"
    },{
      "Guid":"0477875f-4c6b-4a98-823e-ba044420ca09","Name":"DP-M-
2016STD","Id":8,"Suffix":"","IpAddress":"[redacted]","MacAddress":"[redacted]44
:AF","Version":"8.5.0.195","Enabled":true,"Status":"Online","InstallDate":"2016-
10-26T18:26:12.703+01:00","LastContactDate":"2017-02-09T13:24:37.297Z"
    }
  ]
}
```

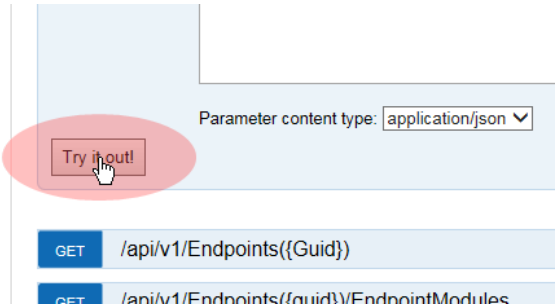

If you're using Internet Explorer, you will be prompted to download the file.



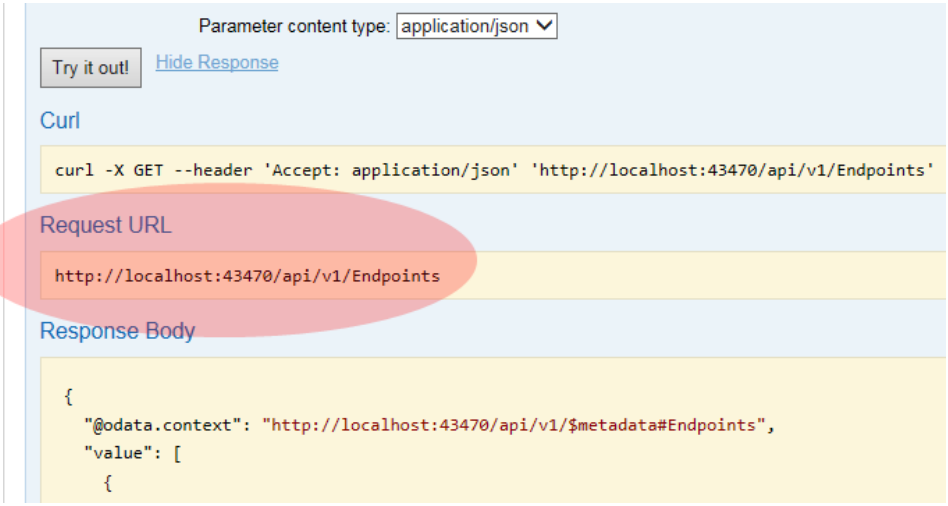
You can also get the above URL from the Swagger™ docs:

1. Expand **Endpoints > GET /api/v1/Endpoints**.

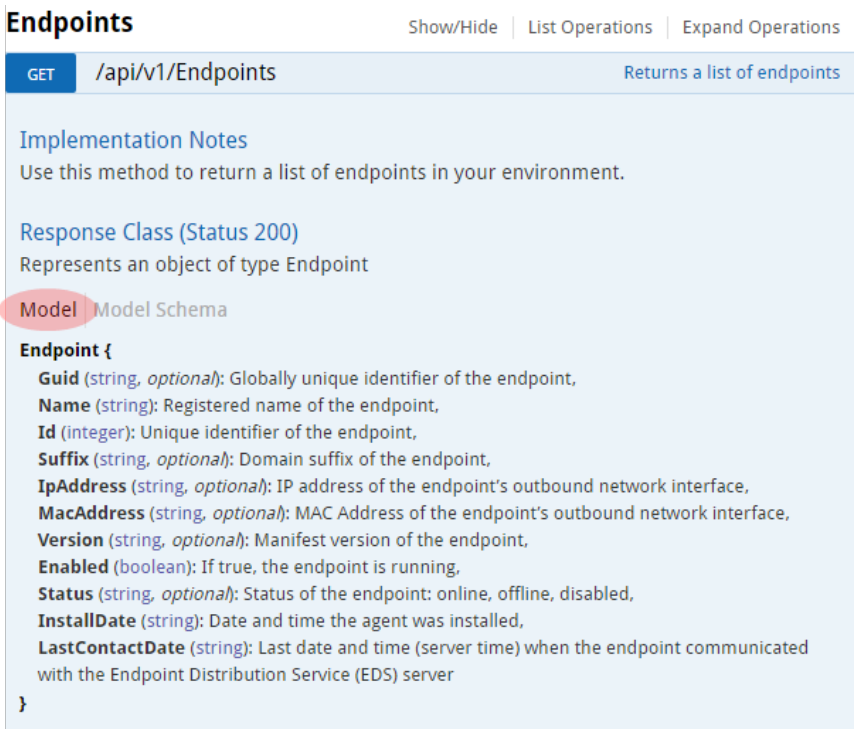
2. Click **Try it out!**



It appears in the **Request URL** field.



Click **Model** to view descriptions of the parameters returned in the Response Body.



Exercise 2: Getting a single endpoint entity

If you examine the list of endpoints from your previous request you'll see that they each have a GUID(Global Unique Identifier) property. This is the key for the Endpoint entity.


```
{
  "@odata.context":"http://localhost:43470/api/v1/$metadata#Endpoints"
    {
      "Guid":"8a6f1210-0b15-4da1-8ef9-d3b155d31d58","Name":"DP-M-
LATEST2012","Id":3,"Suffix":"","IpAddress":"10.5.184.21","MacAddress":
:1D:46","Version":"8.5.0.136","Enabled":true,"Status":"Online","Instal
-10-06T21:30:02.067+01:00","LastContactDate":"2017-02-09T13:16:54.923Z
    },{
```


Many API operations use entity keys to identify individual entities. Keys are usually integers or unique identifiers (GUIDs).

Copy the GUID for one endpoint from the result of the previous call (without the quotes) and use it in the next call. We'll use the one highlighted above:

```
http://localhost:43470/api/v1/Endpoints(8a6f1210-0b15-4da1-
8ef9-d3b155d31d58)
```

Result:


```
{
        "@odata.context":
"http://localhost:43470/api/v1/$metadata#Endpoints/$entity",
        "guid": "8a6f1210-0b15-4da1-8ef9-d3b155d31d58",
        "Name": "PC123-WIN7",
        "Id": 89,
        "Suffix": "corp.lcl",
        "IpAddress": "111.121.122.341",
        "MacAddress": "12:34:56:78:90:AB",
        "Version": "8.5.0.0",
        "Enabled": false,
        "Status": "Disabled",
        "InstallDate": "2017-01-14T18:08:45.943Z",
        "LastContactDate": "2017-08-03T02:22:03.903+01:00"
                }
```


Exercise 3: Requesting only some of an endpoint entity's properties

OData provides a rich query language for selecting and filtering entity collections using keywords such as select, filter, orderby, top and skip.

You can get more information about OData 4.0 in OData Getting Started and OData Version 4.0 Documentation. We recommend consulting the OData Version 3.0 Documentation, which is better documented at the moment.

If you apply a select query to your previous call you can limit the JSON response to only contain the endpoint's name and IP Address.

This is done by adding a question mark (?) , select query keyword ($select) and then a comma separated list of the properties (Name, IPAddress):

```
http://localhost:43470/api/v1/Endpoints(8a6f1210-0b15-4da1-
8ef9-d3b155d31d58)?$select=Name,IpAddress
```

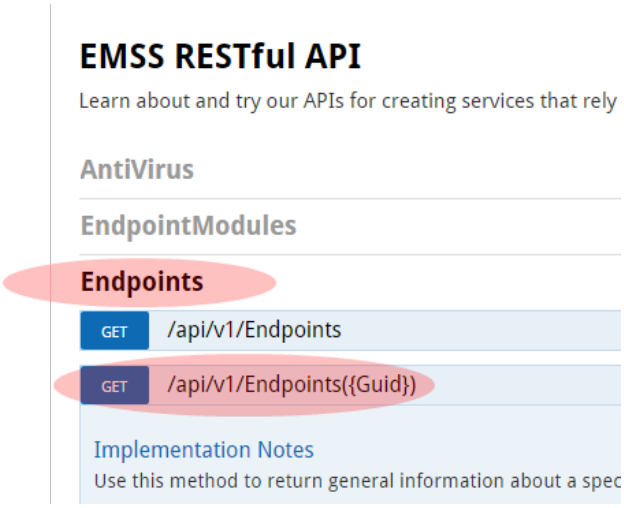
Result:


```
{
"@odata.context": "http://localhost:43470/api/v1/$metadata#Endpoints
(Name,IpAddress)/$entity",
"Name": "DP-M-LATEST2012",
```

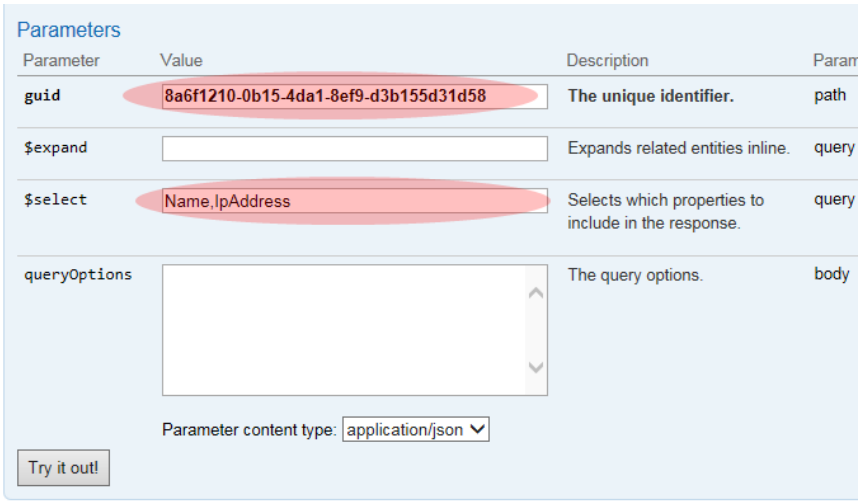
```
"IpAddress": "10.5.184.21"
}
```

Again the Swagger™ docs can help you configure the URL:

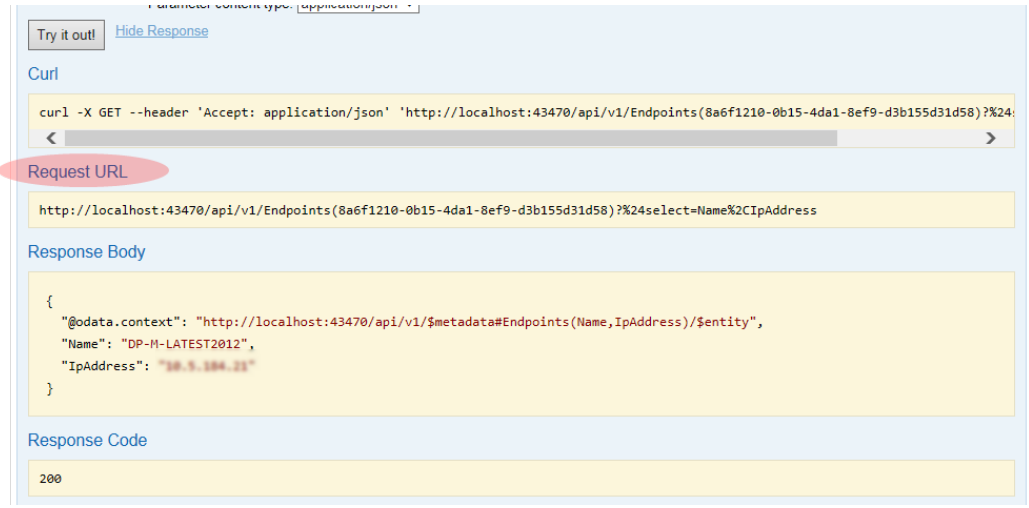
1. Expand **Endpoints > GET /api/v1/Endpoints({Guid})**.



2. In the Parameters section, enter the GUID and $select values.



3. Click **Try it out!**



Exercise 4: Getting a list of Groups

A call to retrieve the Groups entity collection returns all of the groups in the EMSS database including the system and parent groups:

```
http://localhost:43470/api/v1/Groups
```

Exercise 5: Requesting only some group entities by filtering by Group Type

If you apply a filter query to your previous call you can limit the JSON response to only see groups created by a User. This is done by adding a question mark (?), followed by the filer query keyword ($filter=) and then a filter predicate (Type eq 'User'):

```
http://localhost:43470/api/v1/Groups?$filter=Type eq 'User'
```

The first “User” group listed in the JSON result is the parent group for all custom groups. This usually has an Id of 4. If you have not created any custom groups you will only see this group.

Response:

```
{
        "Id": 4,
        "ParentId": 1,
        "Name": "Custom Groups",
        "Type": "User",
```

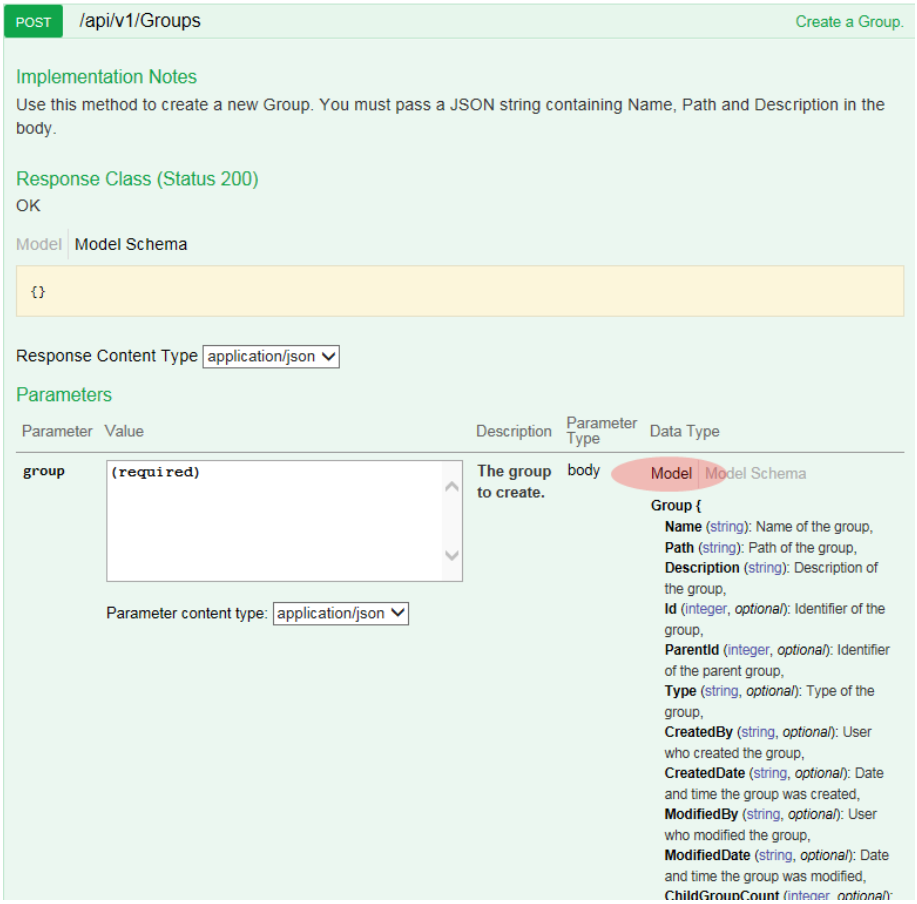
```
        "Path": "OU=Custom Groups,OU=My Groups",
        "Description": "System created parent group to all custom groups",
        "CreatedBy": null,
        "CreatedDate": "0001-01-01T00:00:00Z",
        "ModifiedBy": null,
        "ModifiedDate": "0001-01-01T00:00:00Z",
        "ChildGroupCount": 0,
        "AssignedDeviceCount": 0,
        "AssignedSourceGroupDeviceCount": 0,
        "DerivedDeviceCount": 0,
        "InheritPolicy": false,
        "InheritMandatoryBaseline": false,
        "InheritDeployment": false,
        "MandatoryBaselineEnabled": false,
        "DeploymentEnabled": true,
        "PolicyEnabled": false,
        "QChain": 0
                }
```


Exercise 6: Creating a new Group (a POST request)

You make a new custom group by creating a request that will POST data to the API in the form of JSON. This cannot be done by entering a URL in a browser. It can be achieved using the Swagger™ doc but using a tool with the ability to create URL requests with specific headers and message bodies is recommended.

The Postman extension for Google Chrome is specifically made for this task and is easy to use.

Use the previously returned JSON response (from the Groups request) as a guide to create a new object. Only a few properties are required. Have a look at the parameters in the Swagger™ docs. Click Model to review the Parameter description.



Properties marked as optional are unnecessary. You only need to specify the name, description and where the Group sits in the custom group hierarchy. This is the same as the info required during normal Group creation in the EMSS Web User Interface.

To make the request using Postman fire it up and set the following:

1. Enter the URL: `http://localhost:43470/api/v1/Groups`
2. Select the Request Type **POST**.
3. Select **Body > Raw > JSON (application/json)**

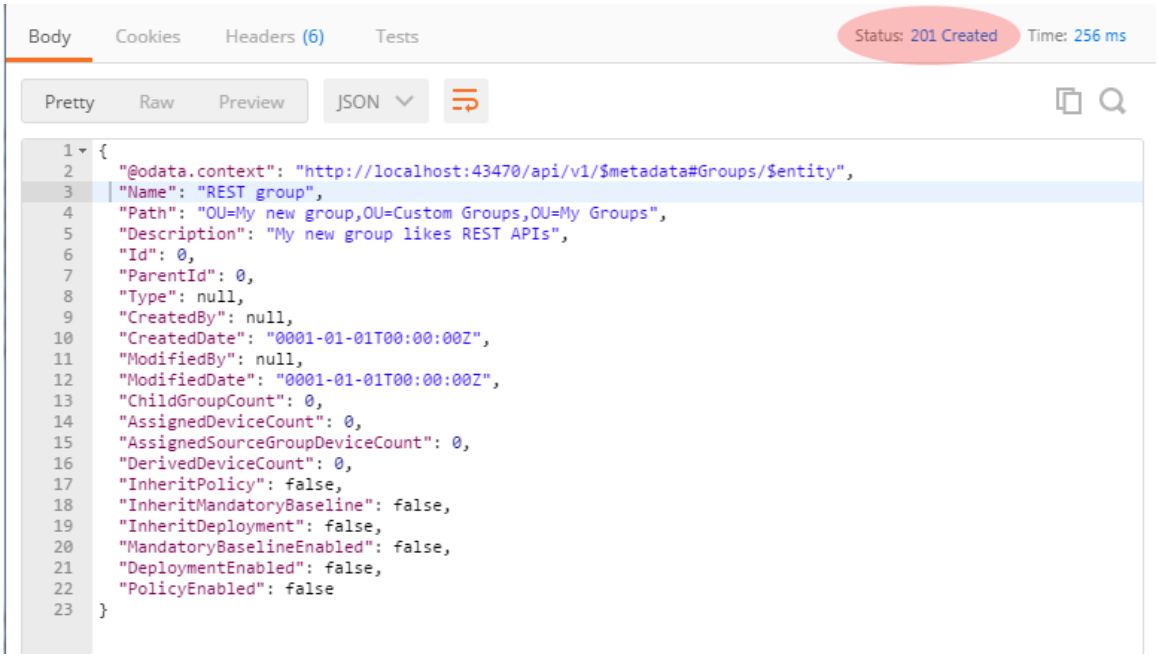
4. Enter the body:

```
{
"Name": "REST group",
"Path": "OU=My new group,OU=Custom Groups,OU=My Groups",
"Description": "My new group likes REST APIs"


}
```

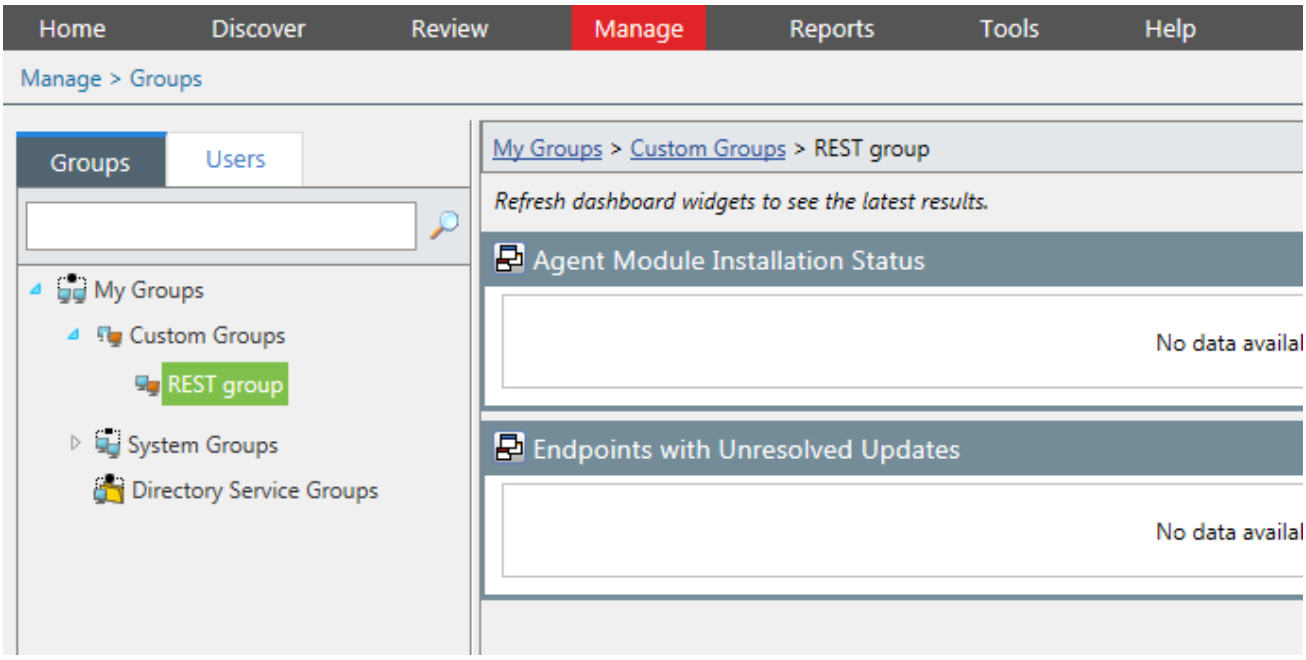
5. Click **Send**.

A response status of “201 Created” to will indicate success.



You can now navigate to the EMSS UI Groups page to confirm your group was created:

1. In EMSS console, select **Manage > Groups**.
2. In the Groups Browser, expand the Custom Groups node.



Exercise 7: Adding an Endpoint to a Group

We can add one or more endpoints to our new custom group. This is another POST request. If you have a look at the Swagger docs you will see that the call takes the form:

```
POST http://localhost:43470/api/v1/Groups({id})/Endpoints
({endpointGuid})
```

You can obtain the necessary group id and endpoint guid using previously discussed API calls.

Exercise 8: Experiment with Endpoint Policies Yourself!

Here's something you can try to prove the usefulness of the Groups API.

Try setting up various custom groups that suit the topology of your endpoint network (e.g. "Finance", "IT" and "Sales") and assign them policies in the EMSS Console for the modules they are installed with.

Now these policies will be applied to the endpoint automatically after the API call is made to add it to the correct group.

Troubleshooting

Top 3 Problems That Cause Issues

1. Installer was not launched using “Run as Administrator”.
2. Not enough user access rights provided during installation.
3. Data entered during installation was not correct:
    - SQL connection string, EMSS License, EMSS user can be reconfigured in: `RESTAPIHost.exe.config`
    - Credentials used for installing/starting service can be fixed by executing `RestAPiHost.exe` from its installation location (default location `C:\Program Files (x86)\HEAT Software\RESTApi`). See http://docs.topshelf-project.com/en/latest/overview/commandline.html

Where are the log files located?

REST API log

Name: HEAT.RESTAPIHost.RESTAPIHost.exe.log

Default location: C:\Program Files\HEAT Software\RESTApi\log

Installer Log

Name: RESTAPI_Install_Log.log

Default location: %temp% (C:\Users\[username]\AppData\Local\Temp)

PUT

Updates a resource and returns HTTP 200 on success.

DELETE

Deletes a resource and returns HTTP 200 on success.

How to upgrade a broken REST API Version 1.0 installation

You have two options:

- Attempt to upgrade the v1.0 instance to make it uninstallable. Then uninstall v1.0 and install v.1.0.1.
- Manually fix the XML configuration file (`RESTAPIHost.exe`) then manually install and start the HEAT REST API service.

Fixing an exiting installation using the v1.0.1 installer:

1. Execute the v1.0.1 installer (Run as Administrator).
2. Complete the upgrade wizard. There is no reconfiguration in an upgrade scenario, so faulty settings from the initial installation will still prevent the REST API service from starting. This step is required only to achieve a successful uninstallation.
3. Open **Control Panel > Add/Remove Programs** then select **HEAT Rest API** and uninstall it.
4. Rerun the v1.0.1 installer.

Fixing an existing installation by manually changing the configuration file:

1. Open a command prompt with Administrator privileges and navigate to the application installation folder.
2. Run `RESTAPIHost.exe uninstall` (you might see errors, that is OK)
3. Navigate to the location where the REST API was installed.
4. Open the `RESTAPIHost.exe.config` XML file.
5. Locate the tag `<connectionStrings></connectionStrings>` and enter the correct data for both Databases PLUSDatabase and UPCCommonDatabase (Data Source and Initial Catalog values).
6. Locate the tag `<appSettings></appSettings>` and enter correct data for:
    - **SSL** true if SSL used false if SSL not used
    - **SSLCretPath** - path to the certificate
    - **licensekey** - license number for EMSS the REST API is going to point to
    - **port** – port number used by REST service
    - **username** – EMSS username used as ContectID to access EMSS SQL database (EMSS User)
7. Open an elevated command line and run `RESTAPIHost.exe`. If all the data entered is correct the service will start from the command line and log to the console.

8. Install the HEAT Rest API application as a Windows service by running the following command:
    - `RESTAPIHost.exe install –networkservice` (if using windows authentication)
    - `RESTAPIHost.exe install –username:DomainServiceAccount –password:password` (if using SQL authentication)