Getting Started with REST API

v2

Copyright Notice

Trademark Legal Notice

Contents

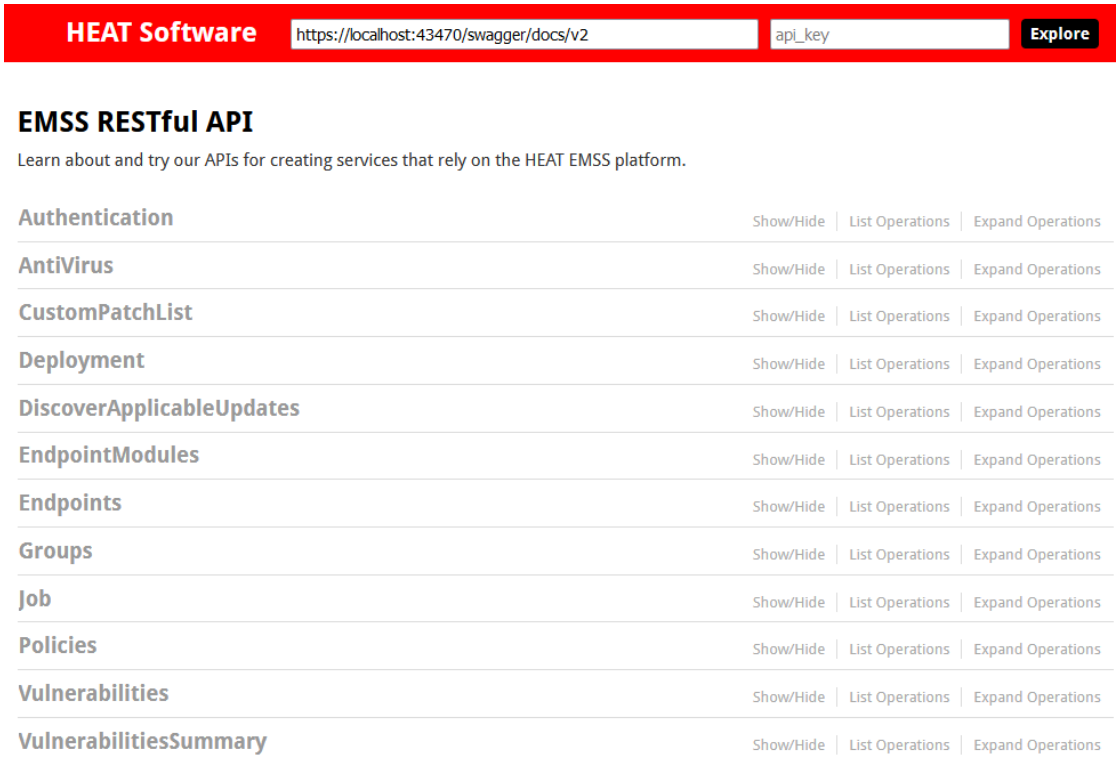
Interactive documentation

The fastest way to learn about the Endpoint Security 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.

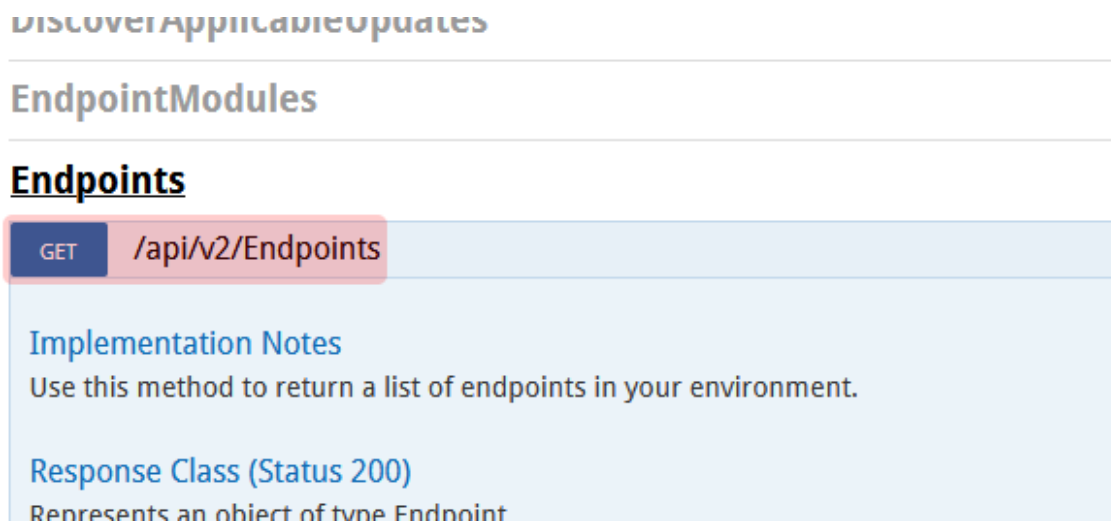
1. Open a browser and enter one of the following URLs:

    - Without SSL: `http://<host>:<port>/docs/index`
    - With SSL: `https://<host>:<port>/docs/index`

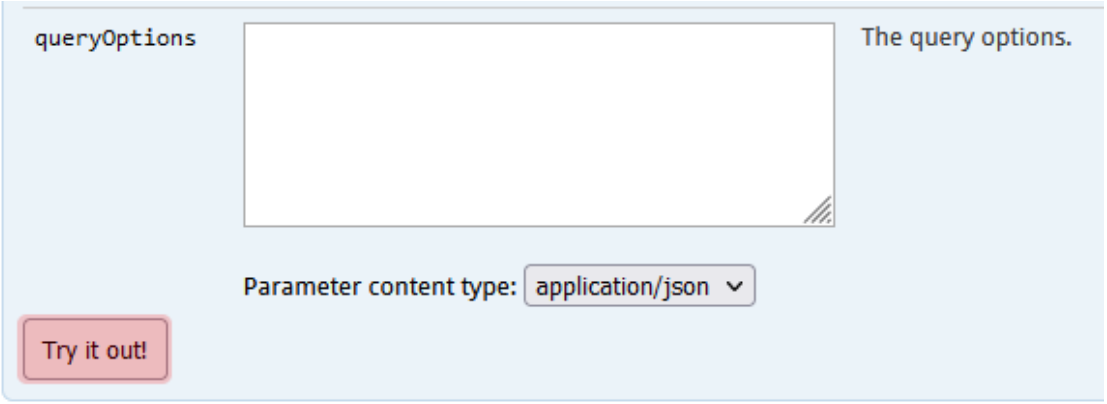
    The HEAT REST API Swagger documentation appears:



2. Expand **Endpoints > GET /api/v2/Endpoints**:



3. Click **Try it out!**



Some API endpoints use nested classes in their Model Schema, which cannot be displayed in the current implementation of Swagger.
For a full reference for these API endpoints, see "Endpoints with Nested Classes" on page 7.

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.

Requests and Responses

Request URLs

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

`http://<localhost>:<port>/api/v2/Endpoints`

'`v2`' 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.

Format

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

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.

Endpoints with Nested Classes

The Endpoint Security REST API v2 includes endpoints where the **Model Schema** contains nested classes. Unfortunately, due to technical limitations, this cannot be documented in the current implementation of Swagger.
Below is the reference description of these API endpoints.

For API endpoints that are not listed below, please refer to Swagger for further details.

Stringified JSON

The code samples below are indicative for the data that must be provided. The actual body of the POST must be stringified JSON.
So if the provided sample is:

```
{
      Name: "Test",
      JobType: 1,
      StartTime: "2022-03-03T11:30:00.000Z",
      DiscoveryTargets:{
         RangeIpTargets:[{StartIp: "192.168.19.100", EndIP:
         "192.168.19.200"}]
      },
}
```

The actual body will be:

```
{
      "value" : "{
         Name: \"Test\",
         JobType: 1,
         StartTime: \"2022-03-03T11:30:00.000Z\",
         DiscoveryTargets:{
            RangeIpTargets:[{StartIp: \"192.168.19.100\", EndIP:
            \"192.168.19.200\"}]
         },
      }"
}
```

API Reference

POST/api/v2/Jobs

Use this API endpoint for creating a job to discover, install and uninstall agents.

Request URL:

`http://<host>:<port>/api/v2/Jobs`

JSON Body

```
{
      Name: "<JobName>",
      JobType: 1,
      StartTime: "2022-03-03T11:30:00.000Z",
      OverwriteExistingAgents: false,
      Credentials: {
         Username: "<WindowsUserName>",
         Password: "<WindowsPassword>",
      },
      Modules: {
         AC: false,
         AV: false,
         DC: false,
         PR: false,
      },
      DiscoveryOptions: {
         PortScanDiscovery: true,
         IcmpDiscovery: true,
         ResolveDnsName: true,
         ResolveMacAddress: true,
         ResolveNetbiosName: true,
         SnmpDiscovery: true,
         VerifyWithPing: true,
         WindowsDiscovery: true,
      },
      DiscoveryTargets: {
         SingleIpTargets: [{
            Ip: "192.168.0.100",
            Timeout: 3000,
            Retry: 3
         }],
         RangeIpTargets: [{
```

```
            StartIp: "192.168.0.100",
            EndIp: "192.168.0.200",
            Timeout: 3000,
            Retry: 3
         }],
         ComputerNameTargets: [{
            ComputerName: "<ComputerName>",
            Timeout: 3000,
            Retry: 3
         }],
         ActiveDirectoryTargets: [{
            CanonicalName: "<CN>",
            OrganizationalUnit: "<OU>",
            Domain: "<DomainName>",
            Username: "<UserName>",
            Password: "<Password>",
            Timeout: 3000,
            Retry: 3
         }]
      },
}
```

Reference

The descriptions below contain several examples with code snippets. For the full **Model Schema** and to see how these snippets fit in, please expand the sample JSON body above.

- `Name` (required):
  Sets the name of the job

- `JobType` (required):
  Sets the job type. Can have one of the following values:

  `1` - Creates a discovery-only job. Equivalent to **Discover -> Assets...**

  `2` - Creates an install job. Equivalent to **Discover -> Assets and Install Agents...**

  `3` - Creates an uninstall job. Equivalent to **Discover -> Assets and Uninstall Agents...**

- `StartTime` (required):
  Sets the start time of the job.
  - If the start time is set in the past, the job will run immediately. Equivalent to the option **Immediate** from the Scan Wizard.
  - If the start time is set in the future, the scheduling will be equivalent to the option **Once** from the Scan Wizard.
- `OverwriteExistingAgents` (optional):
  Specifies whether to overwrite existing agent installs. This option is equivalent to the option **Overwrite the existing agents**, at **Discover > Assets and Install Agents...** on the **Agent Options** page. Can be `true` or `false`.

  If set to true, overwritten agents will lose their history, group membership and policy assignments.
- `Credentials` (optional):
  Sets the Windows credentials. If not set, discovery-only jobs might work, but agent installs and uninstalls will fail with `Access Denied` message.

- `Modules` (optional):
  Specifies which modules to install if the job is an install job. For other job types, this property is ignored.
  If not set, all modules will be set to false by default.
  To set which modules to install, set the equivalent inner properties to true.

  Examples

  - Install only AV:

    ```
    Modules: {
          AV: true
    }
    ```

  - Install only AC and PR:

    ```
    Modules: {
          AC: true,
          PR: true
    }
    ```

  - Install all modules:

    ```
    Modules: {
          AC: true,
          AV: true,
          DC: true,
          PR: true
    }
    ```

- `DiscoveryOptions` (optional)
  Sets how to discover the assets. If this property is not specified, all discovery options will be set to true.

  Examples

  - Only ICMP is true, the rest of the options are false:

    ```
    DiscoveryOptions: {
          IcmpDiscovery: true
    }
    ```

  - All options true, except PortScanDiscovery:

    ```
    DiscoveryOptions: {
          IcmpDiscovery: true,
          ResolveDnsName: true,
          ResolveMacAddress: true,
          ResolveNetbiosName: true,
          SnmpDiscovery: true,
          VerifyWithPing: true,
          WindowsDiscovery: true
    }
    ```

    Because `PortScanDiscovery` is not listed, it is considered `false` by default.

  - All discovery options set to false:

    ```
    DiscoveryOptions: {}
    ```

- `DiscoveryTargets` (required):
  Sets the discovery targets.
  These targets can currently be discovered using four target types:

  - `SingleIpTargets` (optional)
    Specified using class: `Ip`.
  - `RangeIpTargets` (optional)
    Specified using classes: `StartIp` and `EndIp`.
  - `ComputerNameTargets` (optional)
    Specified using class: `ComputerName`.
  - `ActiveDirectoryTargets` (optional)
    Specified using classes: `CanonicalName`, `OrganizationalUnit`, and `Domain`.

  At least one target type must be set.

  All of the discovery targets have two common properties:

  - `Timeout` (in milliseconds; default value = `3000`)
  - `Retry` (default value = `3`)

  If these two properties are missing, the default value will be considered.

  Examples

  - Only one IP address, Timeout is 3000ms and Retry is 3:

    ```
    DiscoveryTargets: {
          SingleIpTargets: [{
             Ip: "192.168.0.1"
          }]
    }
    ```

    Because `Timeout` and `Retry` are not listed, their default values are used.

  - Only one computer name, Timeout is 5000ms and Retry is 3:

    ```
    DiscoveryTargets: {
          ComputerNameTargets: [{
             ComputerName: "my pc",
             Timeout: 5000
          }]
    }
    ```

- One computer name with Timeout 5000ms and Retry 3 and one address with Timeout 3000ms and Retry 10:

```
DiscoveryTargets: {
      ComputerNameTargets: [{
         ComputerName: "my pc",
         Timeout: 5000
      }],
      SingleIpTargets: [{
         Ip: "192.168.0.1",
         Retry: 10
      }]
 }
```

- Multiple single IP addresses:

```
DiscoveryTargets: {
      SingleIpTargets: [{
         Ip: "192.168.0.1"}, {Ip: "192.168.0.2"}, {Ip:
         "192.168.0.2"
      }]
}
```

Examples of invalid configuration

- No target type specified:

```
DiscoveryTargets: {}
```

- Target types specified, but no actual targets specified:

```
DiscoveryTargets: {
      ComputerNameTargets: [],
      SingleIpTargets: []
}
```

POST/api/v2/DeploymentGroups

Use this API endpoint to create a deployment of a custom patch list to a deployment group. This is equivalent to using the Deployment Wizard.

Request URL:

`http://<host>:<port>/api/v2/DeploymentGroups`

JSON Body

```
{
      CustomPatchListId: 1,
      AgentGuids: ["9e317d51-07ac-4c17-8a1f-3111d385bc7e"],
      GroupIds: [1009],
      StartTime: "2023-01-06T11:35:02.007Z",
      Notify: {
         Message: "Notification options",
         AllowCancel: false,
         AllowSnooze: true,
         NotificationOnTop: true,
         UseAgentPolicies: false,
         UseAgentPolicy_AllowCancel: false,
         UseAgentPolicy_AllowSnooze: true,
         UseAgentPolicy_NotificationOnTop: true,
         DeployWithin: 60,
      }
      Reboot: {
         Message: "Reboot options",
         AllowCancel: false,
         AllowSnooze: true,
         NotificationOnTop: true,
         UseAgentPolicies: false,
         UseAgentPolicy_AllowCancel: false,
         UseAgentPolicy_AllowSnooze: true,
         UseAgentPolicy_NotificationOnTop: true,
         RebootWithin: 60,
      }
}
```

Reference

The descriptions below contain several examples with code snippets. For the full **Model Schema** and to see how these snippets fit in, please expand the sample JSON body above.

- `CustomPatchListId` (required):
  The ID of the custom patch list base on which the deployment is created. For more details, please refer to the Custom Patch List documentation.

- `AgentGuids` (optional if `GroupIds` is specified):
  An array containing the target endpoints.

- `GroupIds` (optional if `AgentGuids` is specified):
  An array containing the target groups.

- `StartTime` (required)
  The start time of the deployment.

- `Notify` (optional)
  Equivalent to the **Deployment Notification Options** on the **Notification Options** page of the Deployment Wizard.

  - `Message` (optional)
    The notification message. The REST API does not support localization. so the default message will always be in English. The default value is: "The download and installation of the patch: {Package Name} is ready to begin. If you require any additional information, please contact your Ivanti Endpoint Security administrator."

  - `AllowCancel` (optional; default is `false`)
    Allows the user to cancel the deployment.

  - `AllowSnooze` (optional; default is `true`)
    Allows the user to snooze the deployment.

  - `NotificationOnTop` (optional; default is `true`)
    Specifies if the notification should appear on top.

  - `UseAgentPolicies` (optional; default is `false`)
    Specifies if the deployment should use agent policies.

  - `UseAgentPolicy_AllowCancel` (optional; default is `false`)
    Specifies if the deployment should use agent policies for `AllowCancel`.

  - `UseAgentPolicy_AllowSnooze` (optional; default is `true`)
    Specifies if the deployment should use agent policies for `AllowSnooze`.

  - `UseAgentPolicy_NotificationOnTop` (optional; default is `true`)
    Specifies if the deployment should use agent policies for `NotificationOnTop`.

  - `DeployWithin`
    The time in minutes after which the deployment will begin.

  If `Notify` is not present in the JSON body, endpoints will not be notified of this deployment.
  If `Notify` is present, but no inner options are specified (for example `Notify: {}`), the default values for the notification options, as used in the Deployment Wizard, is applied.
  You can overwrite any individual option by setting it inside the `Notify` JSON structure. For options that are not overwritten, the default value is applied.

- `Reboot` (optional)
  Equivalent to the **Reboot Notification Options** on the **Notification Options** page of the Deployment Wizard.
  This property uses the same options as listed for `Notify` (directly above), except for `DeployWithin`. Instead, it uses:

  - `RebootWithin`
    The time in minutes after which the reboot will begin.

If `Reboot` options are set in the request body, but the deployment does not require a reboot, these options will be ignored. This behavior is identical to the Deployment Wizard: the user cannot set reboot options if the deployment does not require a reboot.

Examples

- No `Notify` or `Reboot` options:

```
{
      CustomPatchListId : 1,
      AgentGuids : ["9e317d51-07ac-4c17-8a1f-3111d385bc7e"],
      GroupIds: [1009],
      StartTime: "2023-01-06T11:35:02.007Z",
}
```

- Overwriting only some of the `Notify` options:

```
{
      CustomPatchListId : 1,
      AgentGuids : ["9e317d51-07ac-4c17-8a1f-3111d385bc7e"],
      GroupIds: [1009],
      StartTime: "2023-01-06T11:35:02.007Z",
      Notify: {
         AllowCancel: true,
         DeployWithin: 222
      }
}
```

- Overwriting some `Notify` and some `Reboot` options:

```
{
      CustomPatchListId : 1,
      AgentGuids : ["9e317d51-07ac-4c17-8a1f-3111d385bc7e"],
      GroupIds: [1009],
      StartTime: "2023-01-06T11:35:02.007Z",
      Notify: {
         AllowCancel: true,
         DeployWithin: 222
      },
      Reboot: {
         AllowSnooze: false,
         RebootWithin: 222
      }
}
```

Use Cases

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

See the "Walkthrough of Endpoints and Groups" on page 31 for information on how to configure a POST request.

Authentication

Before you can start using v2 of the REST API, you need to have to obtain an access token (JWT) using a valid Endpoint Security username and password.

Solutions

- **POST/api/v2/Authentication/Login** to return a JWT to access the API.
  Request URL:
  `http://<host>:<port>/api/v2/Authentication/Login`

Agent Policy Sets

Import, export or update Agent Policy Sets

You need to import, export, update agent policy sets between systems..

Solutions

- **GET /api/v2/AgentPolicySets/HEAT.RESTAPI.Export()** to export the agent policy sets to an XML file.
  Request URL:
  `http://<host>:<port>/api/v2/AgentPolicySets/HEAT.RESTAPI.Export()`

- **POST/api/v2/AgentPolicySets/HEAT.RESTAPI.Import** to read a previously exported XML file and import the policies into IES.
  Request URL:
  `http://<host>:<port>/api/v2/AgentPolicySets/HEAT.RESTAPI.Import`

AntiVirus

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/v2/AvAlerts** to return a list of all alerts generated in your environment.
  Request URL:
  `http://<host>:<port>/api/v2/AvAlerts`

- **GET /api/v2/Endpoints({guid})/AvAlerts** to return the alerts generated by a specific endpoint.
  Request URL:
  `http://<host>:<port>/api/v2/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/v2/Modules** to return a list of modules installed on endpoints.
  Request URL:
  `http://<host>:<port>/api/v2/Modules`

- **GET /api/v2/Endpoints({guid})/EndpointModules** to return a list of modules installed on a specific endpoint.
  Request URL:
  `http://<host>:<port>/api/v2/Endpoints(<Guid>)/EndpointModules`

- **GET /api/v2/Endpoints({guid})/AvDefinition** to return information about the AntiVirus definitions file installed on a specific endpoint.
  Request URL:
  `http://<host>:<port>/api/v2/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/v2/Policies** to get information about the AntiVirus policies in your environment.
  Request URL:
  `http://<host>:<port>/api/v2/Policies`

- **GET /api/v2/Policies{'{PolicyType}'}** to return a list of AntiVirus policies of a specific type.
  Request URL:
  `http://<host>:<port>/api/v2/Policies(<Policy Type>)`

- **GET /api/v2/Endpoints({guid})/AvDefinition** to return information about the AntiVirus definitions file installed on a specific endpoint.
  Request URL:
  `http://<host>:<port>/api/v2/Endpoints(<Guid>)/AvDefinition`

Application Control Policies

Import or export Application Control Policies

You need to import or export Application Control Policies between systems.

Solutions

- **GET /api/v2/Policies/HEAT.RESTAPI.AcPoliciesExport()** to export the Application Control Policies to an XML file.
  Request URL:
  `http://<host>:<port>/api/v2/Policies/HEAT.RESTAPI.AcPoliciesExport()`

- **POST/api/v2/Policies/HEAT.RESTAPI.AcPoliciesImport** to read a previously exported XML file and import the policies into IES.
  Request URL:
  `http://<host>:<port>/api/v2/Policies/HEAT.RESTAPI.AcPoliciesImport`

Discover, install and uninstall agents

Create Jobs to discover, install and uninstall agents

You need to discover, install and uninstall agents. Each of these is done by creating a job.

Solutions

- **POST/api/v2/Jobs** to create a job.
  Request URL: `http://<host>:<port>/api/v2/Jobs`

Custom Patch Lists

Manage Custom Patch Lists

You need to query, create and delete Custom Patch Lists, and add content to them.

Solutions

- **GET/api/v2/CustomPatchLists** to return a list of all custom patch lists in your environment.
  Request URL:
  `http://<host>:<port>/api/v2/CustomPatchLists`
- **GET/api/v2/CustomPatchLists({id})/Bulletins** to return the contents of a specified custom patch list.
  Request URL:
  `http://<host>:<port>/api/v2/CustomPatchLists(<Id>)/Bulletins`
- **POST/api/v2/CustomPatchLists** to create a new, empty custom patch list.
- **PUT/api/v2/CustomPatchLists({Id})/Bulletins** to add bulletins to a custom patch list.
  Request URL:
  `http://<host>:<port>/api/v2/CustomPatchLists(<Id>)/Bulletins`
- **DELETE/api/v2/CustomPatchLists({Id})** to delete a specified custom patch list.
- **DELETE/api/v2/CustomPatchLists({Id})/Bulletins** to delete specified bulletins from a specified custom patch list.

Create a deployment based on a Custom Patch List

You need to deploy a Custom Patch Lists and add content to them.

Solutions

- **GET/api/v2/DeploymentGroup** to return a list of all deployment groups in your environment.
  Request URL:
  `http://<host>:<port>/api/v2/DeploymentGroup`
- **POST/api/v2/DeploymentGroup** to create the deployment of a custom patch list to a deployment group.

Query the status of deployment tasks and deployment targets

You want to query deployment tasks and targets.

Solutions

- **GET/api/v2/DeploymentGroup({Id})/DeploymentTasks** to return all deployment tasks of a specified deployment group.
  Request URL:
  `http://<host>:<port>/api/v2/DeploymentGroup({Id})/DeploymentTasks`

- **GET/api/v2/DeploymentGroup({Id})/DeploymentTasks({guid})/DeploymentTargets** to get the deployment targets for a deployment task.

Groups

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/v2/Groups** to return a list of all the groups in your environment.
  Request URL:
  `http://<host>:<port>/api/v2/Groups`
- **GET /api/v2/Groups({Id})** to return information about a specific group.
  Request URL:
  `http://<host>:<port>/api/v2/Groups(<Id>)`
- **GET /api/v2/Groups({id})/Endpoints** to return information about the endpoints in a specific group.
  Request URL:
  `http://<host>:<port>/api/v2/Groups(<Id>)/Endpoints`
- **POST /api/v2/Groups** to create a new group.
- **POST /api/v2/Groups({id})/Endpoints({endpointGuid})** to add specific endpoints to a specific group.
- **DELETE /api/v2/Groups** to delete an existing group.
- **DELETE /api/v2/Groups({id})/Endpoints({endpointGuid})** to delete specific endpoints in a specific group.

Endpoints

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/v2/Modules** with an OData filter to get a list of modules where the value of the parameter `IsPendingReboot` is `True`.
  Request URL:
  `http://<host>:<port>/api/v2/Modules?$filter=IsPendingReboo t eq true`

- **GET /api/v2/Endpoints({guid})/EndpointModules** to get a list of modules installed on a specific endpoint and information about them. Check the status of the parameter `IsPendingReboot` for `AntiVirus`.
  Request URL:
  `http://<host>:<port>/api/v2/Endpoints(<Guid>)/EndpointModules`

System Synchronization

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

Solution

- **GET /api/v2/Endpoints** to get a list of endpoints in your environment.
  Request URL:
  `http://<host>:<port>/api/v2/Endpoints`

Update the Display name for an Endpoint

You want to update the **Display Name** in Endpoint Security for an endpoint in your environment.

Solution

- **PUT put /api/v2/Endpoints({guid})/HEAT.RESTAPI.UpdateDisplayName** to update the **Display Name** of the specified endpoint.
  Request URL:
  `http://<host>:<port>/api/v2/Endpoints(<Guid>)/HEAT.RESTAPI.UpdateDisplayName`

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/v2/Endpoints({guid})/Vulnerability** to return vulnerabilities for a specific endpoint.
  Request URL:
  `http://<host>:<port>/api/v2/Endpoints(<Guid>)/Vulnerability`

- **GET /api/v2/VulnerabilitiesSummary** to return a summary of the vulnerabilities patched/not patched in your environment.
  Request URL:
  `http://<host>:<port>/api/v2/VulnerabilitiesSummary`

- **GET /api/v2/VulnerabilitiesSummary({EndpointGuid})** to return a summary of the vulnerabilities patched/not patched on a specific endpoint.
  Request URL:
  `http://<host>:<port>/api/v2/VulnerabilitiesSummary(<EndpointGuid>)`

Mandatory Baselines

Manage the content of mandatory baselines for groups.

You need to manage the mandatory baseline to ensure these are up to date and contain all relevant elements.

Solution

- **GET /api/v2/Groups({Id})/MandatoryBaselines** to return the content of the mandatory baseline for a specified group.
  Request URL:
  `http://<host>:<port>/api/v2/Groups(<Id>)/MandatoryBaselines`

- **POST/api/v2/Groups({Id})/MandatoryBaselines** to add content to the mandatory baseline for a specified group.
  Request URL:
  `http://<host>:<port>/api/v2/Groups(<Id>)/MandatoryBaselines`

- **DELETE/api/v2/Groups({Id})/MandatoryBaselines** to delete content from the mandatory baseline for a specified group.
  Request URL:
  `http://<host>:<port>/api/v2/Groups(<Id>)/MandatoryBaselines`

DAU Scans

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/v2/Endpoints({guid})/Vulnerability** to return information about a DAU scan on a specific endpoint.
  Request URL:
  `http://<host>:<port>/api/v2/Endpoints(<Guid>)/Vulnerability`

- **POST/api/v2/DiscoverApplicableUpdates/HEAT.RESTAPI.ScanNow** to schedule a DAU scan to be performed as soon as possible on specified agents and/or groups.
  Request URL:
  `http://<host>:<port>/api/v2/DiscoverApplicableUpdates/HEAT.RESTAPI.ScanNow`

Walkthrough of Endpoints and Groups

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:

- Ivanti Endpoint Security (formerly HEAT 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/v2/` .
Your root URL will depend on the port you chose during installation and whether you are accessing the API remotely. In this walkthrough we'll use localhost and (default) port 43470.

When accessing the REST API remotely, ensure you have access to the port chosen during installation (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.

Authentication

The following endpoint can be used to obtain a JWT Token:

`http://localhost:43470/api/v2/Authentication/Login`

JSON Request Body

```
{
      "Username": "<IES_username>",
      "Password": "<IES_password>"
}
```

JSON Response

```
{
      "Jwt": "<jwt>"
}
```

The JWT Expires after 15 minutes, a new JWT must be generated for further usage.

Exercise 1: Get the entity collection of installed endpoints

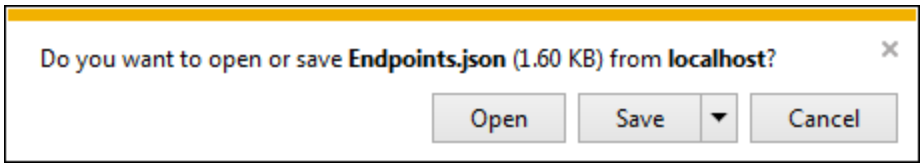
First let's test you have everything setup correctly by requesting a list of the endpoints in your environment. Enter this call into your browser:

```
http://localhost:43470/api/v2/Endpoints
```

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

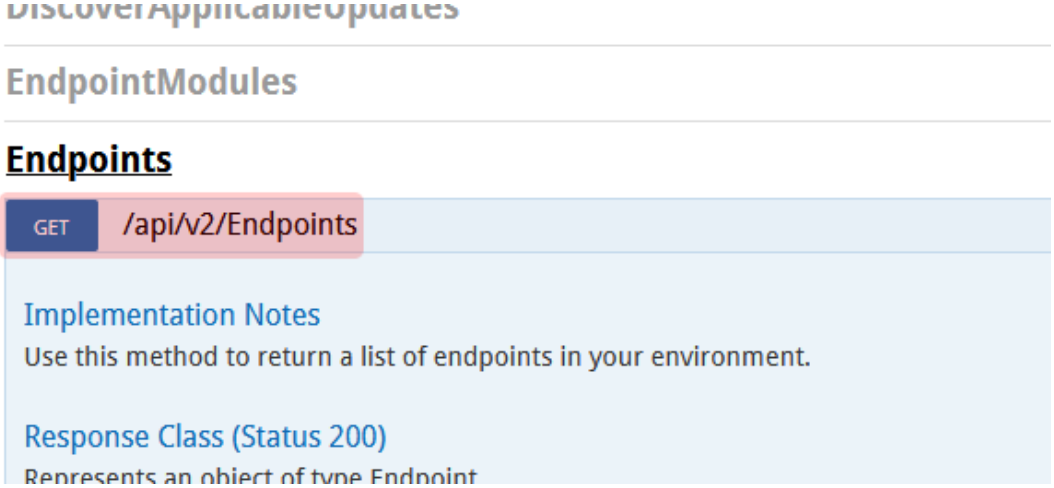

```
{
  "@odata.context":"http://localhost:43470/api/v1/$metadata#Endpoints","value":[
    {
      "Guid":"8a6f1210-0b15-4da1-8ef9-d3b155d31d58","Name":"SRV2019-
FS01","Id":1,"Suffix":"example.local","IpAddress":"10.38.29.27","MacAddress":"FD:F1:
3B:44:CA:8F","Version":"8.6.0.10","Enabled":TRUE,"Status":"Online","InstallDate":"20
20-12-04T10:15:18.55Z","LastContactDate":"2020-12-22T11:12:42.88Z"
    },{
      "Guid":"6d420193-1f8a-4585-b1a4-3840d12504d6","Name":"SRV2019-
FS02","Id":2,"Suffix":"example.local","IpAddress":"10.38.24.180","MacAddress":"FD:F1
:3B:44:CB:9F","Version":"8.6.0.10","Enabled":TRUE,"Status":"Online","InstallDate":"2
020-12-04T10:18:15.47Z","LastContactDate":"2020-12-22T11:18:32.854Z"
    },{
      "Guid":"140dbc7f-9a8a-4a3f-a561-4180a1772782","Name":"W10x64-
DT0123","Id":5,"Suffix":"example.local","IpAddress":"10.38.39.29","MacAddress":"A4:9
A:20:00:41:A7","Version":"8.6.0.10","Enabled":FALSE,"Status":"Offline","InstallDate"
:"2020-12-06T11:05:18.45Z","LastContactDate":"2020-12-08T10:02:26.832Z"
    },{
      "Guid":"74a2eb8c-bd98-41f7-9674-6801d0a1e99b","Name":"W10x64-
DT0142","Id":6,"Suffix":"example.local","IpAddress":"10.38.39.38","MacAddress":"7B:E
E:ED:DD:19:E5","Version":"8.6.0.10","Enabled":TRUE,"Status":"Online","InstallDate":"
2020-12-08T10:22:23.95Z","LastContactDate":"2020-12-21T11:52:47.4Z"
    },{
      "Guid":"1b2d640f-b2d9-4bef-9916-5561f0c041fe","Name":"W10x64-
LT0342","Id":8,"Suffix":"example.local","IpAddress":"10.38.39.31","MacAddress":"EB:4
3:29:61:F6:08","Version":"8.6.0.10","Enabled":TRUE,"Status":"Online","InstallDate":"
2020-12-07T09:27:34.22Z","LastContactDate":"2020-12-22T11:12:47.951Z"
    }
  ]
}
```


Depending on the browser you use, you may be prompted to download the file.



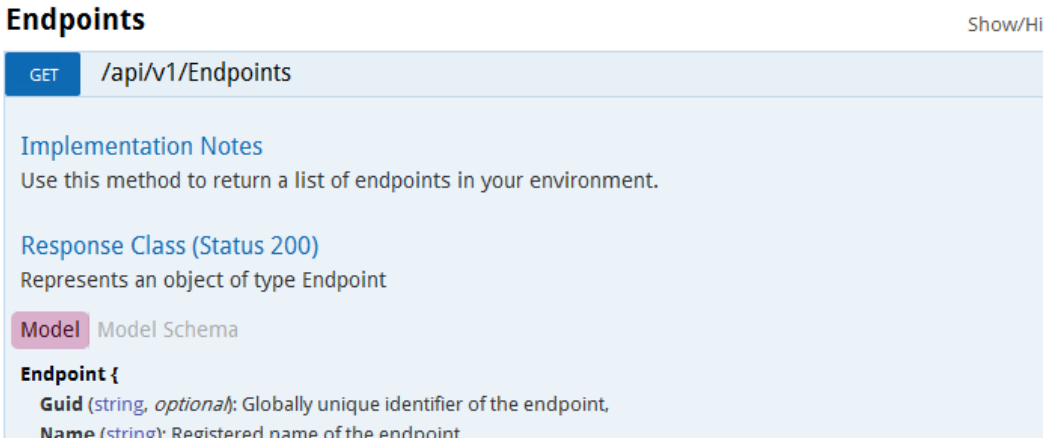
You can also get the above URL from the Swagger docs:

1. Expand **Endpoints > GET /api/v2/Endpoints**.



2. Click **Try it out!**
   The URL appears in the **Request URL** field.

3. 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","value":[
    {
      "Guid":"8a6f1210-0b15-4da1-8ef9-d3b155d31d58","Name":"SRV2019-
FS01","Id":1,"Suffix":"example.local","IpAddress":"10.38.29.27","MacAddress":"FD:F1:
3B:44:CA:8F","Version":"8.6.0.10","Enabled":TRUE,"Status":"Online","InstallDate":"20
20-12-04T10:15:18.55Z","LastContactDate":"2020-12-22T11:12:42.88Z"
    },{
```


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/v2/Endpoints(8a6f1210-0b15-4da1-8ef9-d3b155d31d58)`

JSON-response, with line breaks inserted for clarity


```
{
      "@odata.context":
      "http://localhost:43470/api/v2/$metadata#Endpoints/$entity",
      "guid": "8a6f1210-0b15-4da1-8ef9-d3b155d31d58",
      "Name": "SRV2019-FS01",
      "Id": 1,
      "Suffix": "example.local",
      "IpAddress": "10.38.29.27",
      "MacAddress": "FD:F1:3B:44:CA:8F",
      "Version": "8.6.0.10",
      "Enabled": true,
      "Status": "Online",
      "InstallDate": "2020-12-14T10:15:18.55Z",
      "LastContactDate": "2020-12-22T11:12:42.88Z"
            }
```

Exercise 3: Requesting only some of the properties of an endpoint

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 ODataVersion 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`):

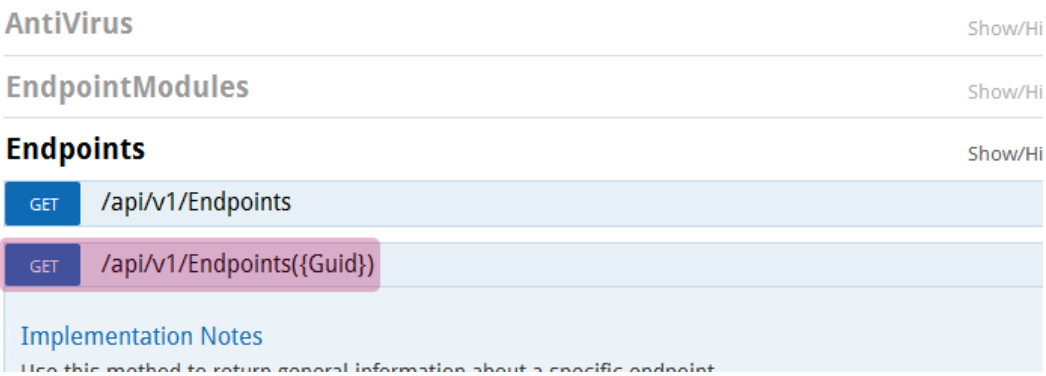
```
http://localhost:43470/api/v2/Endpoints(8a6f1210-0b15-4da1-8ef9-d3b155d31d58)?$select=Name,IpAddress
```

JSON-response, with line breaks inserted for clarity:

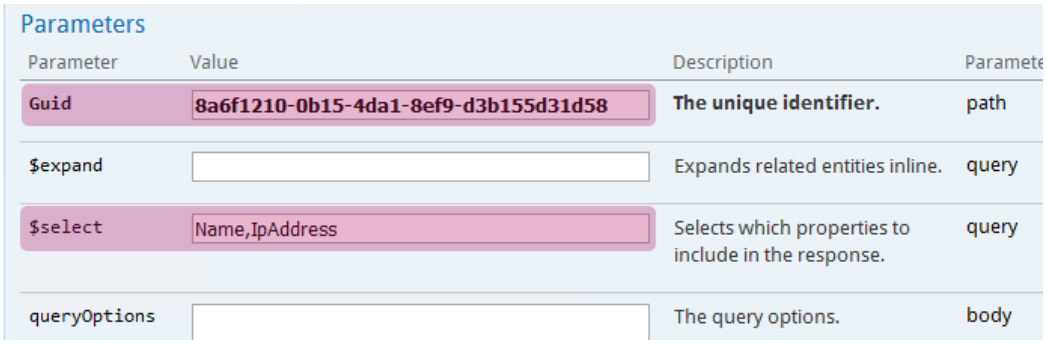
```
{
      "@odata.context": "http://localhost:43470/api/v2/$metadata#Endpoints
      (Name,IpAddress)/$entity",
      "Name": "SRV2019-FS01",
      "IpAddress": "10.38.29.27"
            }
```

Again the Swagger docs can help you configure the URL:

1. Expand **Endpoints > GET /api/v2/Endpoints({Guid})**.



2. In the **Parameters** section, enter the values for **Guid** and **$select**.



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 IvantiEndpoint Security database, including the system and parent groups:

```
http://localhost:43470/api/v2/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 filter query keyword (`$filter=`) and then a filter predicate (`Type eq 'User'`):

```
http://localhost:43470/api/v2/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.

JSON-response, with line breaks inserted for clarity:

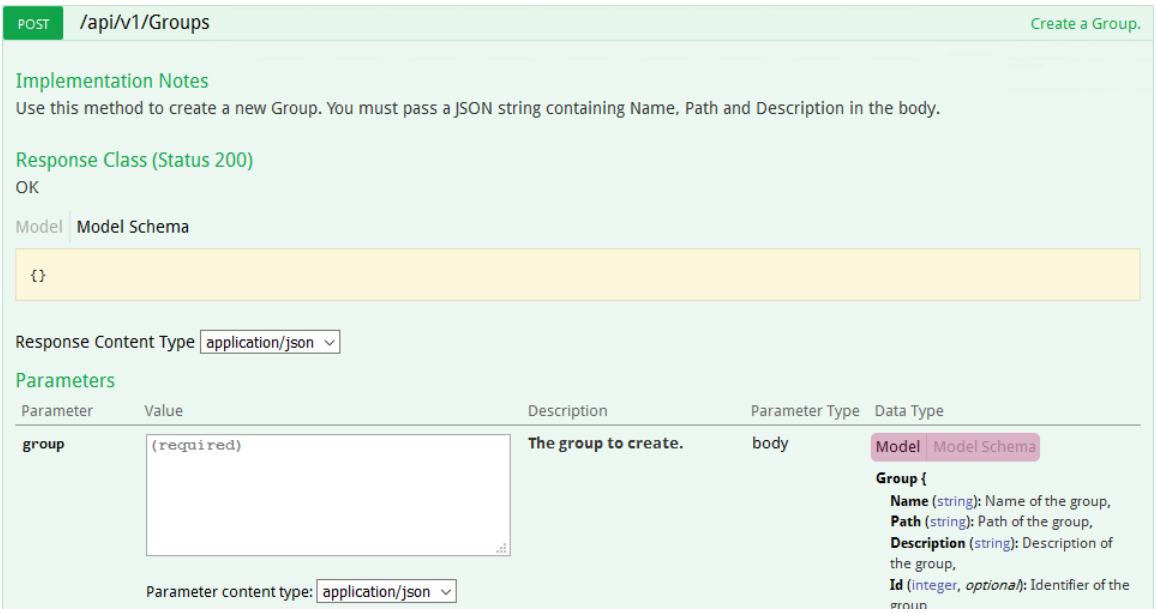

```
{
      "Id": 4,
      "ParentId": 1,
      "Name": "Custom Groups",
      "Type": "User",
      "Path": "OU=Custom Groups,OU=My Groups",
      "Description": "System created parent group to all custom groups",
      "CreatedBy": null,
      "CreatedDate": "0001-01-01T00:00:00Z",
      "ModifiedBy": 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 JSON-response from the Groups request as a guide to create a new object.
Have a look at the parameters in the Swagger docs. Click **Model** to review the descriptions of the properties that can be specified in the groups parameter.



You must specify the name, description and where the Group sits in the custom group hierarchy. This is the same information as is required during normal Group creation in the Endpoint Security Web User Interface.
The rest of the properties are marked as optional.

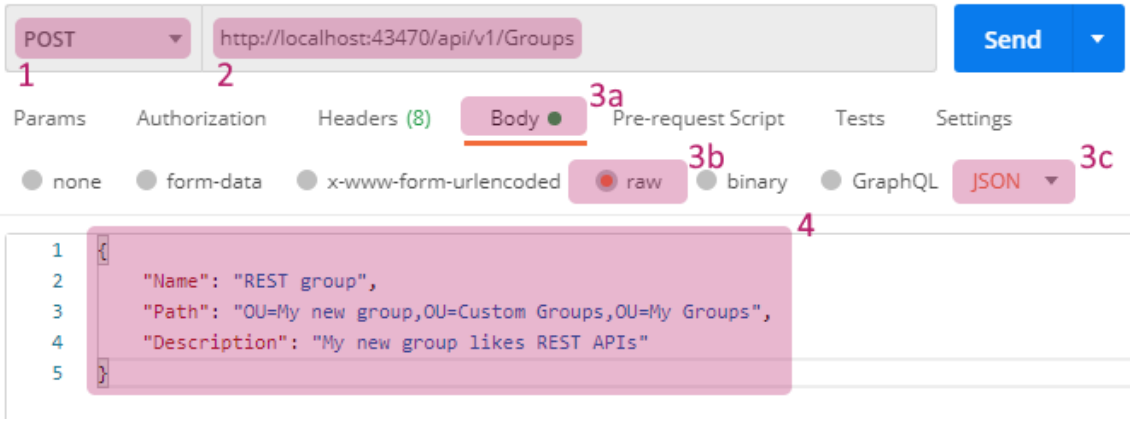
To make the request using Postman, set the following in the Postman app:

1. Select the Request Type **POST**.
2. Enter the URL: `http://localhost:43470/api/v2/Groups`
3. On the **Body** tab (a), select **raw** (b) and **JSON** (c).

4. Enter the body:

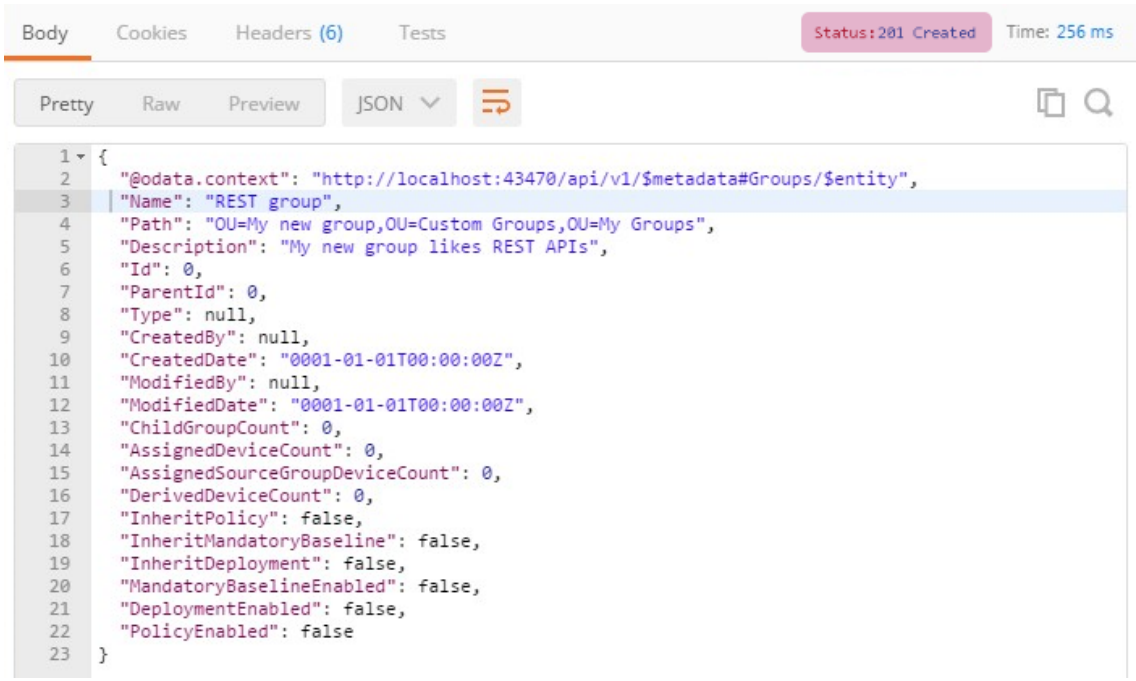
```
{
        "Name": "REST group",
        "Path": ,
         "Description": "My new group likes REST APIs"
                }
```

The result of steps 1 through 4 should look like this:



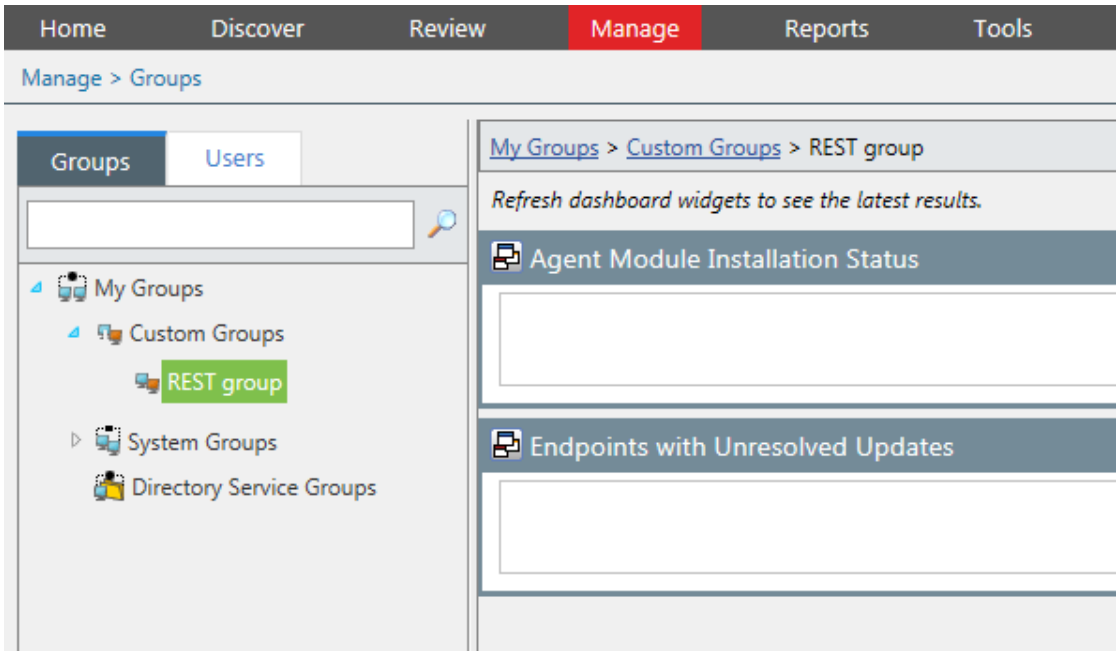
5. Click **Send**.

A response status of **201 Created** to will indicate success.



You can now navigate to the IvantiEndpoint Security**Groups** page to confirm your group was created:

1. In the Endpoint Security Console, go to **Manage > Groups**.
2. In the **Groups** browser, expand the **Custom Groups** node.



Exercise 7: Adding an Endpoint to a Group

You can add one or more endpoints to your 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/v2/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 Endpoint Security 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, Endpoint Security License, Endpoint Security 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(x86)\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:

Fixing an existing installation using the v1.0.1 installer

Attempt to upgrade the v1.0 instance to make it uninstallable, then uninstall v1.0 and install REST API v2:

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 > Programs and Features**, then select **HEAT RestAPI** and uninstall it.
4. Run the installer for REST API v2.

Fixing an existing installation by manually changing the configuration file

Manually fix the XML configuration file (`RESTAPIHost.exe`) then manually install and start the HEAT REST API service:

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
    - `SSLCertPath` - path to the certificate
    - `licensekey` - license number for the Endpoint Security instance the REST API is going to point to
    - `port` – port number used by REST service
    - `username` – Endpoint Security username used as ContextID to access the Endpoint Security SQL database (Endpoint Security 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 RestAPI 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)