Splunk® SOAR (On-premises)
REST API Reference for Splunk SOAR (On-premises) 5.4.0

Generated: 10/28/2022 9:30 pm

Table of Contents

Table of Contents

Table of Contents

Using the Splunk SOAR REST API

Using the REST API reference for Splunk SOAR (On-premises)

The Splunk SOAR (On-premises) platform supports RESTful APIs in order to create, update, and selectively remove objects from the system.

Authentication

REST API requests must be performed over HTTPS, and only authorized users and devices are allowed. User and token based authentication methods exist.

Some REST API calls require user based authentication, for example, deleting records. HTTP Basic auth for user based authentication can be easily performed by the Python requests module:

```
requests.get('https://192.168.1.1/rest/container/1', auth=('admin', 'password'))
```

Authentication can also be provided in the URL:

```
https://username:password@192.168.1.1/rest/container/1
curl -u "username:password" https://192.168.1.1/rest/container/1
```

For token based authentication, the token can be provided in the URL, or `ph-auth-token` must be present in the HTTP headers. Using the token in the password field of the request with no username allows rest access without requiring a valid Splunk SOAR (On-premises) user. See the sample Python script below in the "Provisioning an Authorization Token" section for an example using requests with `ph-auth-token` in the HTTP headers.

```
curl -u ":authToken" https://192.168.1.1/rest/container/1
```

Provisioning an authorization token

Use the `automation` user provided in Splunk SOAR (On-premises) by default to acquire an authorization token. This user and any other automation type users are service accounts that provide access to the REST API with customizable restrictions.

1. Log in to Splunk SOAR (On-premises) as an administrative user.
2. From the Main Menu, select **Administration**.
3. Select **User Management > Users**.
4. Click **+ User** to add a new user.
5. Select **Automation** as the user type.
6. Provide a user name and fill in the Allowed IPs. Use **any** for unrestricted access, a single IP or a single netmask.
7. Choose one or more roles for the new user. The default Automation role is provided for this purpose and has a broad set of permissions that allows most activities that a service account might need. If you wish to have a more restricted set of permissions for a certain playbook or activity, create a role with the desired permissions and assign that instead.
8. Click **Create**.

You can view the token and other configuration information in the Authentication Configuration for REST API panel by clicking on the user name you just created. Cut and paste this JSON formatted data and provide it to your script or application that will be sending the REST requests. Below is some example JSON data:

```
{
  "ph-auth-token": "cs76HmsNcWjkd6kWmGzUa18LcbtQx95vMW1bsdeP7gU=",
  "server": "https://172.16.210.137"
}
```

The configuration contains:

Parameter	Datatype	Description
ph-auth-token	String	Contains the generated authorization token. This token is only valid from the associated IP address provided in the USER EDIT panel.
server	String/URL	Contains the URL that can be used to POST to this Splunk SOAR (On-premises) instance. Provided for convenience.

If the token is compromised or needs to be re-provisioned at any point. You can click on the RE-GENERATE AUTH TOKEN button. A new token will be provided and the old token will no longer be accepted.

Example Python script

This is a simple script that shows how to take a CSV file that represents some activity and uses it to generate containers which will be called "incidents" and artifacts which will be called "events" in Splunk SOAR (On-premises).

Data used for this example:

```
"IP ADDRESS",PORT,SHA256,SCORE,COMMENT,"INDICATOR ID"
1.1.1.1,22,dc4c065cd7618b508857246f8243922253aad50a9943c6e206027db11919bcf4,99,"hack attempt",12387
10.10.10.15,80,3e4ebded4ee802790e465893f17fbc2c456426804381a44b343ae1152e13ebbe,5,"normal operation", 87492
10.10.10.25,443,4a6b9635eae00157b7b38a5a92c23df01df90b656bb61450497086cf074d3f89,20,"bad certificate", 19
1.2.3.4,80,e8d8711bf846af1580ae390da7e6722633ff187d976246b547df9eac25ac5a43,10,"http vpn",7373642
```

The made-up data above contains some standard fields that are expected from various technologies, such as IP address, port and a SHA256 hash, as well as some fields that are custom and specific to the product producing them such as "score" and "comment". The above snippet of data can be put into a file called "example.csv". The following Python program will read the file and create one incident with an artifact per line.

```
# This example uses the 3rd party "requests" module.  This can be installed
# with "pip install requests" from your client machine. If requests is not
# available, any library which supports https POSTS and basic authentication
# can be used.
import os, sys, csv
import json
import requests

AUTH_TOKEN = "tuI6TaoiBv3fjtFcuQLKciCY+niZ87C2l4FLWcWQf7I="
PHANTOM_SERVER = "172.16.131.154"
ARTIFACT_LABEL = "event"

headers = {
  "ph-auth-token": AUTH_TOKEN
}

container_common = {
    "description" : "Test container added via REST API call",
}

# disable certificate warnings for self signed certificates
requests.packages.urllib3.disable_warnings()

def add_container(name, sid):
  url = 'https://{}/rest/container'.format(PHANTOM_SERVER)

  post_data = container_common.copy()
  post_data['name'] = '{} ({})'.format(name, sid)
  post_data['source_data_identifier'] = sid
  json_blob = json.dumps(post_data)

  # set verify to False when running with a self signed certificate
  r = requests.post(url, data=json_blob, headers=headers, verify=False)
  if (r is None or r.status_code != 200):
    if r is None:
      print('error adding container')
    else:
      print('error {} {}'.format(r.status_code,json.loads(r.text)['message']))
    return False

  return r.json().get('id')

def add_artifact(row, container_id):
  url = 'https://{}/rest/artifact'.format(PHANTOM_SERVER)

  post_data = {}
  post_data['name'] = 'artifact for {}'.format(row[4])
  post_data['label'] = ARTIFACT_LABEL
  post_data['container_id'] = container_id
  post_data['source_data_identifier'] = "source data primary key for artifact or other identifier"

  # The cef key is intended for structured data that can be used when
  # dealing with product agnostic apps or playbooks. Place any standard
  # CEF fields here.
  cef = {
    'sourceAddress': row[0],
    'sourcePort': row[1],
    'hash': row[2],
  }

  # The "data" key can contain arbitrary json data. This is useful for
  # keeping data that does not comfortably fit into CEF fields or is highly
  # product specific
  data = cef.copy()
  data['score'] = row[3]
  data['comment'] = row[4]

  post_data['cef'] = cef
  post_data['data'] = data

  json_blob = json.dumps(post_data)

  # set verify to False when running with a self signed certificate
  r = requests.post(url, data=json_blob, headers=headers, verify=False)

  if (r is None or r.status_code != 200):
    if (r is None):
      print('error adding artifact')
    else:
      error = json.loads(r.text)
      print('error {} {}'.format(r.status_code, error['message']))
    return False

  resp_data = r.json()
  return resp_data.get('id')

def load_data(filename):
  with open(filename, 'rb') as csvfile:
    reader = csv.reader(csvfile)
    first_row = True
    for row in reader:
      if first_row:
        # skip the header
        first_row = False
        continue
      if not row:
       continue

      container_id = add_container(row[4], row[5])
      if not container_id:
        continue
      print 'added container {}'.format(container_id)
      artifact_id = add_artifact(row, container_id)

if __name__ == '__main__':
  if len(sys.argv) < 2:
    print "Filename is required"
    sys.exit(-1)

  load_data(sys.argv[1])
  sys.exit(0)
```

Checking for existing records using REST

The above code is very simple and does not cover all of the available options when working with the Splunk SOAR (On-premises) REST API. For example, the csv file above contains an "INDICATOR ID" which is the ID of the high level structure in the source product. It is the closest thing to an incident in that product and more than one row in the csv file may be associated with a specific ID. The same relationship can be preserved when adding data with the REST API. In order to do this, you will want to query to see if a particular document already exists in your Splunk SOAR (On-premises) system. The following code snippet shows how to do this.

```
indicator_id = row[5]
query_url = "https://{}/rest/container".format(PHANTOM_SERVER_IP)
query_url += "?_filter_source_data_identifier=\"{}\"".format(indicator_id)
query_url += "&page_size=1"
response = requests.get(query_url, headers=headers, verify=False)
if (response is None):
    print "Query failed."
elif response.status_code != 200:
    print 'Error: code {}, content: {}'.format(response.status_code, response.text)
else:
  resp_data = response.json()
  num_records = resp_data.get('count')
  if num_records:
    # document already exists, no need to insert
    incident_id = resp_data['data'][0]['id']
  else:
    # document does not exist, it must be added before continuing.
```

```
...
```

Now the example.csv might contain some duplicate INDICATOR IDs.

```
"IP ADDRESS",PORT,SHA256,SCORE,COMMENT,"INDICATOR ID"
1.1.1.1,22,dc4c065cd7618b508857246f8243922253aad50a9943c6e206027db11919bcf4,99,"hack attempt",12387
10.10.10.15,80,3e4ebded4ee802790e465893f17fbc2c456426804381a44b343ae1152e13ebbe,5,"normal operation", 87492
10.10.10.25,443,4a6b9635eae00157b7b38a5a92c23df01df90b656bb61450497086cf074d3f89,20,"bad certificate", 19
1.2.3.4,80,e8d8711bf846af1580ae390da7e6722633ff187d976246b547df9eac25ac5a43,10,"http vpn",7373642
10.10.10.200,80,c8885d33c29e9b7f2278b39afb3ebddb8de526143ca32f7f1c618dc8841a5983,90,"data
exfiltration",12387
10.10.10.105,22,f4adf8a8f6eaff961887f0076ce5080902599e25ba14421e3e2e47f5c990d876,5,"authorized
access",7373642
```

The above code is using the filter feature of the REST API that allows for simple queries to be run. The HTTP GET request above is asking for any one container record which has a "source_data_identifier" field with a particular value. If the count returned is 0, that means so such record exists and a new one should be created before adding the artifact data. If a record does exist, then its' ID has been returned and it can be used when creating the artifact record. This ensures that the one-to-many relationship between containers and artifacts is preserved.

Setting severity and sensitivity of data

Another common requirement is to set the severity and sensitivity of containers and artifacts. Both containers and artifacts have a severity which can be set to "low", "medium" or "high". Containers may also have their sensitivity set to "white", "green", "amber" or "red" as specified in the Traffic Light Protocol. The following Python snippet shows some possible scenarios.

```
# This applies to containers and artifacts. For this example company policy
# states that a score of 75 or higher is a high severity event and 50 or
# higher is a medium severity event.
score = int(row[3])
if score >= 75:
  post_data['severity'] = 'high'
elif score >= 50:
  post_data['severity'] = 'medium'
else:
  post_data['severity'] = 'low'

# This applies only to containers. containers for VPN events are to be
# have limited distribution while access to data exfiltration events requires
# a higher level of privilege.

comment = row[4]
if comment == 'data exfiltration':
  post_data['sensitivity'] = 'red'
elif 'vpn' in comment:
  post_data['sensitivity'] = 'amber'
else:
  post_data['sensitivity'] = 'green'
```

Posting over REST and automation

By default, POSTing a new artifact will trigger automation on the associated container. However, this can be controlled by setting the "run_automation" parameter in your POST JSON. Setting this to false will prevent automation from running after the artifact is added. This happens on a per-container basis. Typically a script ingesting multiple artifacts per container in a batch would turn off automation on all artifacts except the last one which would then cause automation to be run on the fully populated container. A pseudo-code example follows:

```
# Acquire some container/artifact data to post
container = get_container()
artifacts = get_artifact_list_for_container(container)

# post your container
do_post_container(container)
# Set run_automation to false on all but the last one
for artifact in artifacts[:-1]:
  artifact["run_automation"] = False
  do_post_artifact(artifact)
do_post_artifact(artifacts[-1])
```

The "run_automation" flag is also available when POSTing new containers. However it is defaulted to false and it is typically not necessary to modify this since running automation on a container usually requires artifact data.

Query for Data

Splunk SOAR (On-premises) allows authorized users to use a REST API to query for several kinds of information. Including any Containers and Artifacts in the system.

General Form for a Query

Querying for information is done using a HTTP GET request. The response will contain JSON data in the body. The result may be a Javascript Array or a single Javascript Object depending on what was queried. The format of the URL follows:

```
/rest/<type>/[identifier]/[detail]?page=0&page_size=10&pretty&_filter_XXX='YYY'&_exclude
_XXX='YYY'&include_expensive
```

Parameter	Required	Description
type	required	The type of data being queried. Supported types are: • action_run • artifact • asset • app • app_run • container • playbook_run • cluster_node Querying for only a type will return a paginated array of objects.
identifier	optional	Adding an identifier to a URL will retrieve a single specific object. Identifiers are integers and are unique for each resource.
detail	optional	Adding a detail can only be done when querying a single object. The result is to return information on a single field of the object.
page	optional	Positive integer. Returned results are paginated. This query parameter requests a specific page.
page_size	optional	

Parameter	Required	Description
		Positive integer. Returned results are paginated. This query parameter determines how many results returned per-page. Use "0" for all results. Setting page_size to 0 will return all results. Querying a very large data set can have an adverse effect on performance.

prettyoptionalNo value. Adding ?pretty (or &pretty) to your query will add related or calculated fields prefixed with _pretty_ to the response. For instance if the record you are looking for has a start_time, the response will have _pretty_start time that is a relative local version of the time. A record that has an "owner" reporting back an id will gain _pretty_owner that shows the owner's display name.

Requesting pretty values is a relatively expensive call since it may involve expensive calculations or additional database lookups. When querying individual records or small numbers of records, this should not cause any performance hit. However if requesting tens of thousands of records it may have an adverse impact on your system depending on your hardware.

_filter_XXXoptionalAdd one or more filters to limit the results. Applies only to lists of objects. See Filtering below._exclude_XXXoptionalExclude matching items with syntax similar to filtering. See Excluding below.include_expensiveoptionalNo value. Adding this flag will cause the REST API to return all fields when returning a list, including large/expensive fields.

The include_expensive parameter will return all fields, just as if you were requesting the individual record. These expensive fields may have megabytes of data for a single record, so use this option with caution as it may have a significant performance impact if returning large amounts of data.

sortoptionalField to sort results with. Can be any "simple" field at the top-level of record, such as a string, boolean, or integer value that is not under a hierarchy. Custom fields for events can also be sorted, using the format custom_fields.field_name.orderstringEither "asc" or "desc". This is the sorting order for the results.

Requesting a Single Object

The following is a query returning a single `Container`. The returned value is a single JSON object.

```
Request URL:
/rest/container/42

Response body:
{
    "artifact_count": 14,
    "artifact_update_time": "2015-11-24T22:45:09.185206+00:00",
    "asset": 101,
    ...
    "start_time": "2014-10-19T14:40:33+00:00",
    "status": "closed",
    "tags": [""],
    "version": 1
}
```

Requesting an Array of Objects

This is an example of querying for all `Artifact` objects. The result is a JSON Object containing 3 fields.

Field	Type	Description
count	Integer	Count of total number of records.
num_pages	Integer	Total number of pages available for this query.
data	Array	

Field	Type	Description
		Array of objects in JSON format. The paginated result of the query. Queries that return a list of objects do not include all data. Some fields may be excessively large and are only returned when a single object is queried. One example is the Artifact data field.

```
Request URL:
/rest/artifact?page=2

Response body:
{
    "count": 40,
    "data": [
        {
            "asset": 11,
            "container_id": 5,
            "create_time": "2014-10-19T12:41:33",
            ...
            "start_time": "2014-10-19T14:41:33",
            "type": "network",
            "version": "1.0"
        },
        {
            "asset": 12,
            "container_id": 5,
            "create_time": "2014-10-19T12:40:33",
            ...
            "start_time": "2014-10-19T14:40:33",
            "type": "network",
            "version": "1.0"
        },
        ...
        {
            "asset": 20,
            "container_id": 7,
            "create_time": "2014-09-04T12:40:33",
            "end_time": null,
            ...
            "start_time": "2014-09-04T14:40:33",
            "type": "",
            "version": "1.0"
        }
    ],
    "num_pages": 4
}
```

Filtering

You can add one or more filters that will limit the results you get back on your query. Filters are added as query string parameters and have the following format:

```
/rest/endpoint?_filter_<field_name>=<value>
```

Field name can be anything in the top level of the record you are querying. Using the example provided, you could limit the artifacts to only network artifacts by adding `&_filter_type="network"`.

Depending on your filter, you may want to pass various different kinds of data as the value. You can pass strings, numbers, and simple data structures.

Type	Format	Example
string	Enclosed in double or single quotes.	"myvalue"
number	no formatting	10
array	Python syntax for a literal list.	["a string", 2, None]

When filtering for a status, use the status' id, which is a number, rather than its name, which is a string. You can get the status' id using /rest/container_status.

For more information about /rest/container_status, see REST Status.

You can do more than simple comparisons with filters. You can do substring comparisons, inclusion in a collection and relative values for example. You could do this by adding an operator to the filter. For instance to change the above example to a substring search I can do `/rest/artifact?_filter_type__contains="net"`.

A case-insensitive version would be `/rest/artifact?_filter_type__icontains="net"`.

You can also do some limited filtering on fields of related records. For instance if I wanted to filter artifacts for artifacts belonging to containers that have "spyware" in the name, I could use `/rest/artifact?_filter_container__name__icontains="spyware"`.

The filter API resembles the Django filter syntax. You can see documentation on the various operators features available on the Django queryset filter documentation.

You can add multiple filters to a request. However they will be ANDed together. There is no way to do an OR between filters. Not all features of Django querysets are supported.

Excluding

The opposite of filtering is excluding items that match certain criteria. You can use a similar syntax in excluding as you can use in filtering. The simplest format for filtering is the following:

```
/rest/endpoint?_exclude_<field_name>=<value>
```

If you want to query for artifacts that do not have the tag `preprocessed`, use the following URL:

```
/rest/artifact?_exclude_tags__contains="preprocessed"
```

See Filtering for more complicated syntax you can use, such as a substring comparison or inclusion in a list. All operators supported for filtering are also supported for excluding.

Requesting Object Detail

The simplest query for object detail to retrieve a field such as the `Artifact name` or `Container close_time` etc. The result is a JSON object with a key/value pair for the data queried.

```
Request URL:
/rest/container/5/name

Response body:
```

```
{
    "name": "CryptoLocker Ransomware Infection (new, SLA breached) (192.168.1.41)"
}
```

A more useful operation is to find objects that are related to a principle object. Such as querying for all actions run in the context of a Container. To facilitate getting the detail information on such queries, pseudo fields are available giving a object list of these related objects.

```
Request URL:
/rest/container/5/actions?page=1

Response body:
{
    "count": 18,
    "data": [
        {
            "action": "url reputation",
            "assign_time": "2015-03-21T21:03:57.728000",
            "cancelled": null,
            "close_time": "2015-03-21T21:04:00.425000",
            ...
            "type": "investigate",
            "update_time": "2015-03-21T21:04:00.425000",
            "version": "1.0"
        },
        {
            "action": "timeliner",
            "assign_time": "2015-03-21T21:00:12.796000",
            "cancelled": null,
            "close_time": "2015-03-21T21:02:46.212000",
            ...
            "update_time": "2015-03-21T21:02:46.212000",
            "version": "1.0"
        },
        ...
        {
            "action": "snapshot vm",
            "assign_time": "2015-03-21T20:57:21.944000",
            "cancelled": null,
            "close_time": "2015-03-21T20:58:01.897000",
            ...
            "update_time": "2015-03-21T20:58:01.897000",
            "version": "1.0"
        }
    ],
    "num_pages": 2
}
```

These are the currently defined pseudo fields.

Type	Pseudo field	Description
action_run	app_runs	App runs (app commands) created by the Action.
action_run	approvals	Approvals associated with the Action.
app_run	log	Get logs (from spawn.log) pertaining to an app run id. Debug Logging has to be enabled.
container	actions	Actions created (both automated and manual) in response to the Container.
container	attachments	Any files attached to the Container (vault items).

Type	Pseudo field	Description
container	playbook_runs	Playbooks run (both automated and manual) in response to the Container.
container	artifacts	All Artifacts that contribute to the Container.
container	audit	Audit information for a container.
container	comments	Comments added to the container.
container	recommended_actions	Brief set of recommendations for appropriate actions for the container.
container	recommended_playbooks	Brief set of recommendations for appropriate playbooks for the container.
playbook	audit	Audit information for a Playbook.
playbook_run	actions	Actions created by the Playbook.
playbook_run	log	Logged messages from playbook run (logging must have been enabled).
ph_user	roles	Roles that the user belongs to.
ph_user	audit	Audit information for the user.
role	users	Users that belong to the role.
role	audit	Audit information for users that belong to the role.

Update Records

It is possible to update existing records using REST. Complete or partial updates can be made by doing an HTTP POST to the record ID that you want to update.

/rest/<type>/<id>

Syntax

```
https://<username>:<password>@<host>/rest/<type>/<id>
```
POST

Update an existing container.

Example request
Update container Id 10 with a new name and severity.

```
curl -k -u admin:changeme https://localhost/rest/container/10 \
-d '[
{
    "name": "my new container name",
    "severity": "low"
}
]'
```
Example success response
A successful POST will return the success message and Id.

```
{
    "id": 10,
    "success": true
}
```

Example failure response
A failed POST will return the failure message and reason.

```
{
    "failed": true,
    "message": "<reason>"
}
```

Any record that can be created using the REST API can be updated as described.

Bulk Create and Update Records

Records can be created and updated in bulk. The bulk create and bulk update operations behave differently than you might expect.

The bulk operations do not perform transactional edits. A transactional edit means that either all of the updates succeed or none succeed.

The result of a bulk API call can be that half of the operations succeed and half fail, but the return is an HTTP 200 success status code if at least one operation succeeds.

The bulk operations return a list of the response bodies that are generated, as if the client had called the create or update API many times.

Therefore, the following scenario is possible:

1. You send a bulk update request to modify 100 records.
2. All but one of the records fails.
3. You get an HTTP 200 response.
4. You don't check the individual statuses of each update operation.
5. You think that all is well, but you don't realize that 99 records did not get updated.

One example of a bulk operation is creating bulk container notes.

Example request
The following example request is missing a required parameter in the JSON body for container Id 4.

```
curl -k -u admin:password https://127.0.0.1:8443/rest/note \
-d '[{
                "container_id": 1,
                "phase": 2,
                "author_id": 1,
                "title": "example1",
                "note_type": "general",
                "content": "hello world"
        },
        {
                "container_id": 2,
                "phase": 2,
                "author_id": 1,
                "title": "example2",
                "note_type": "general",
                "content": "hello world"
        },
        {
                "container_id": 3,
                "phase": 2,
                "author_id": 1,
                "title": "example3",
                "note_type": "general",
                "content": "hello world"
        },
        {
                "container_id": 4,
                "phase": 2,
                "author_id": 1,
                "title": "example4"
        }
]'
```

Example response

The body of the following example response shows the success messages for the Ids of the newly created notes, with the exception of one failure.

```
[{
        "id": 4,
        "success": true
}, {
        "id": 5,
        "success": true
}, {
        "id": 6,
        "success": true
}, {
        "failed": true,
        "message": "Missing required parameter: note_type"
}]
```

See /rest/note for further information about notes.

Verify the response body of any bulk operation to make sure that all records are created or updated.

Delete Records

Records can be deleted by an authorized user. This can be done by issuing an HTTP DELETE to `/rest/<type>/<id>`. Deletion can only be done by a user account with the correct privileges and not by a connection authenticated using a REST token. The Python requests module implements both HTTP Basic user auth as well as HTTP DELETE.

/rest/<type>/<id>

Syntax

```
https://<username>:<password>@<host>/rest/<type>/<id>
```

DELETE

Delete a record.

Example Python request
Delete container Id 42.

```
requests.delete('https://192.168.1.1/rest/container/42', auth=('admin', 'password'))
```

Example response
A successful DELETE will return a success message.

```
{
    "success": true
}
```

Example curl request
Delete the custom CEF Id 151.

```
curl -k -u admin:changeme https://localhost/rest/cef/151 -X DELETE
```

Example response
A successful DELETE will return the success message.

```
{
    "success": true
}
```

Failures will return a non-200 response code and JSON with "failed" = true and an appropriate "message".

Delete is only supported for the following record types:

- Apps (/rest/app/<id>)
- Artifacts (/rest/artifact/<id>)
- Assets (/rest/asset/<id>)
- CEF (/rest/cef/<id>)
- Containers (/rest/container/<id>)
- Container Attachments (vault files such as /rest/container_attachment/<id>)
- Custom Lists (/rest/decided_list/<id>)

Custom Lists can be deleted with user or token authentication. All others require user authentication.

Get System Info

This is a set of REST APIs that return information about the system. These endpoints are read-only and don't take POST requests. Access to these APIs requires authentication, but doesn't require special permissions.

/rest/version

Syntax

```
https://<username>:<password>@<host>/rest/version
```

GET

Determine the Splunk SOAR (On-premises) product version.

Example request

```
curl -k -u uname:pwd https://localhost/rest/version -G -X GET
```

Example response

A successful GET returns the version.

```
{
    "version": "X.Y.ZZZ"
}
```

/rest/system_info

Syntax

```
https://<username>:<password>@<host>/rest/system_info
```

GET

Returns the timezone of your instance and the base URL configured in the system settings user interface for your Splunk SOAR (On-premises) instance. The base URL can be blank if it's not configured.

Example request

```
curl -k -u uname:pwd https://localhost/rest/system_info -G -X GET
```

Example response

A successful GET returns the base url and timezone of the Splunk SOAR (On-premises) instance.

```
{
    "base_url": "https://phantom.example.com",
    "time_zone": "UTC"
}
```

/rest/license

Syntax

```
https://<username>:<password>@<host>/rest/license
```

GET

Returns Splunk SOAR (On-premises) license key configuration values and current usage counters. A License Info value of "0" indicates unlimited.

Example request

```
curl -k -u uname:pwd https://localhost/rest/license -G -X GET
```

Example response
A successful GET returns the status and license information.

```
{
        "status": "valid",
        "license_info": {
                "case_management": true,
                "issue_date": 1490227200000,
                "maximum_apps": 0,
                "expiration_date": 1521763200000,
                "maximum_users": 0,
                "contact_information": "soc@example.com",
                "company_name": "example",
                "maximum_assets": 0,
                "license_number": "6bcff42b-4988-4951-9efa-371364102b11",
                "license_type": "standard",
                "maximum_actions_per_day": 0,
                "maximum_containers": 0
        },
        "current_usage": {
                "recent_playbook_run_count": 40,
                "recent_app_run_count": 60,
                "recent_debug_run_count": 60
        }
}
```

/rest/health

Syntax

```
https://<username>:<password>@<host>/rest/health
```
GET

Get memory and CPU usage information for Splunk SOAR (On-premises) services and the overall system.

Example request

```
curl -k -u uname:pwd https://localhost/rest/health -G -X GET
```
Example response
A successful GET returns the status and license information.

```
{
    "all_running": true,
    "db_data": [
        {
            "label": "Free",
            "value": 95413
        },
        {
            "label": "Used",
            "value": 128
        }
    ],
    "last_snapshot_time": "2017-01-28T01:24:40.751000Z",
    "load_data": [
    {
        "load": 0.07,
        "time": "1 minute"
    },
    {
        "load": 0.03,
        "time": "5 minutes"
    },
    {
        "load": 0.0,
        "time": "15 minutes"
    }
],
"memory_data": [
    {
        "label": "Free",
        "value": 3276
    },
    {
        "label": "Used",
        "value": 403
    },
    {
        "label": "Cached",
        "value": 152
    }
],
"services": {
    "actiond": [
        {
            "id": 638,
            "name": "actiond",
            "num_instances": 1,
            "pid": 2465,
            "rs_kb": 2604,
            "stime": 0,
            "time": "2017-01-28T01:24:40.741000Z",
            "utime": 0,
            "vm_kb": 308000
        },
        {
            "id": 631,
            "name": "actiond",
            "num_instances": 1,
            "pid": 2465,
            "rs_kb": 2604,
            "stime": 0,
            "time": "2017-01-28T01:24:10.722000Z",
            "utime": 0,
            "vm_kb": 308000
        },
                ...
            ],
    "decided": [
        {
            "id": 639,
            "name": "decided",
            "num_instances": 1,
            "pid": 2421,
            "rs_kb": 30500,
            "stime": 6,
            "time": "2017-01-28T01:24:40.744000Z",
                "utime": 47,
                "vm_kb": 510932
            },
        ],
        "nginx": [
            {
                "id": 643,
                "name": "nginx",
                "num_instances": 10,
                "pid": 0,
                "rs_kb": 201196,
                "stime": 112,
                "time": "2017-01-28T01:24:40.750000Z",
                "utime": 470,
                "vm_kb": 1363432
            },
        ],
        "ingestd": [
            {
                "id": 641,
                "name": "ingestd",
                "num_instances": 1,
                "pid": 2399,
                "rs_kb": 2776,
                "stime": 0,
                "time": "2017-01-28T01:24:40.747000Z",
                "utime": 0,
                "vm_kb": 162592
            },
        ],
        "postgres": [
            ...
        ],
        "workflowd": [
                        ...
        ]
    },
    "status": {
        "actiond": "running",
        "decided": "running",
        "nginx": "running",
        "ingestd": "running",
        "postgres": "running",
        "watchdogd": "running",
        "workflowd": "running"
    },
    "swap_data": [
        {
            "label": "Free",
            "value": 3967
        },
        {
            "label": "Used",
            "value": 0
        }
    ],
    "vault_data": [
        {
            "label": "Free",
            "value": 95481
        },
        {
            "label": "Used",
            "value": 59
        }
    ]
}
```
The return values of note follow:

Key	Type	Description
all_running	boolean	True if all services are running.
db_data	JSON array	Contains information on free and used space on the /data partition. Values in megabytes.
last_snapshot_time	timestamp	Time when last snapshot was taken in UTC. Snapshots taken every 30 seconds (can vary slightly depending on system load).
load_data	JSON array	Load average information as seen when running "uptime" command.
memory_data	JSON array	Contains 3 values: Used, Free, and Cached memory. Values in megabytes. Does not count swap space.
services	JSON object	One key each for phantom_actiond, phantom_decided, phantom_ingestd, phantom_workflowd, nginx and postgres. Detail infomation below.
status	JSON object	One key each for phantom_actiond, phantom_decided, phantom_ingestd, phantom_workflowd, nginx and postgres. Values either "running" or "stopped" depending on status of each service.
swap_data	JSON array	Contains information on free and used swap space. Values in megabytes.
vault_data	JSON array	Contains information on free and used space on the /opt/phantom/vault partition. Values in megabytes.

The service snapshot details follow:

Key	Type	Description
id	integer	Internal ID of snapshot.
name	string	Name of service.
num_instances	integer	Number of processes for this service.
pid	integer	Process ID of the services (0 for multi-process services).
rs_kb	integer	KB of resident memory used by all instances of the service.
stime	integer	System time used by all instances of the service (in clock ticks). (Kernel processor time)
time	timestamp	Time when snapshot was taken.
utime	integer	User time used by all instances of the service (in clock ticks). (Non kernel processor time)
kb_vm	integer	KB of virtual memory used by all instances of the service.

/rest/app_status

Syntax

```
https://<username>:<password>@<host>/rest/app_status
```
GET

Get test connectivity status for each asset.

Example request

```
curl -k -u uname:pwd https://localhost/rest/app_status -G -X GET
```

Example response

A successful GET returns a default paging size of 10. In the example shown, there are 93 assets, so it will take 10 pages. As with other REST API calls, you can request `?page=N` to get a particular page, or `?page_size=N` to request a different number of items per page.

```
{
  "count": 93,
  "data": [
    {
      "app": 104,
      "asset": 4,
      "create_time": "2017-07-15T21:20:45.113000Z",
      "id": 1,
      "message": "Testing asset connectivity for 'domainctrl1' using app 'LDAP' ",
      "status": "success"
    },
    {
      "app": 57,
      "asset": 5,
      "create_time": "2017-07-15T21:20:50.888000Z",
      "id": 2,
      "message": "Testing asset connectivity for 'alexa' using app 'Alexa' ",
      "status": "success"
    },
    {
      "app": 12,
      "asset": 7,
      "create_time": "2017-07-15T21:21:01.300000Z",
      "id": 3,
      "message": "Testing asset connectivity for 'archer' using app 'RSA Archer' ",
      "status": "success"
    },
    {
      "app": 4,
      "asset": 8,
      "create_time": "2017-07-15T21:21:06.620000Z",
      "id": 4,
      "message": "Testing asset connectivity for 'arcsight_esm' using app 'ArcSight ESM' 1 action succeeded.
",
      "status": "success"
    },
    {
      "app": 88,
      "asset": 9,
      "create_time": "2017-07-15T21:21:16.851000Z",
      "id": 5,
      "message": "Testing asset connectivity for 'ciscoasa' using app 'Cisco ASA' Could not establish ssh
connection to Cisco IOS device. Error reading SSH protocol banner. Connectivity test failed. No action
executions found.. Connectivity test failed",
      "status": "failed"
    },
    {
      "app": 11,
      "asset": 10,
      "create_time": "2017-07-15T21:21:27.048000Z",
      "id": 6,
      "message": "Testing asset connectivity for 'autofocus' using app 'AutoFocus' 1 action failed. Making a
request to PAN AutoFocus. handle_action exception occurred. Error string: 'session_post() takes at least 1
argument (0 given)'",
      "status": "failed"
    },
    {
      "app": 26,
      "asset": 11,
      "create_time": "2017-07-15T21:21:32.626000Z",
      "id": 7,
      "message": "Testing asset connectivity for 'mas' using app 'Malware Analysis Service' 1 action
succeeded. ",
      "status": "success"
    },
    {
      "app": 26,
      "asset": 12,
      "create_time": "2017-07-15T21:21:37.861000Z",
      "id": 8,
      "message": "Testing asset connectivity for 'mas_apikey' using app 'Malware Analysis Service' 1 action
succeeded. ",
      "status": "success"
    },
    {
      "app": 21,
      "asset": 13,
      "create_time": "2017-07-15T21:21:43.067000Z",
      "id": 9,
      "message": "Testing asset connectivity for 'bmcremedy' using app 'BMC Remedy' ",
      "status": "success"
    },
    {
      "app": 63,
      "asset": 14,
      "create_time": "2017-07-15T21:21:48.298000Z",
      "id": 10,
      "message": "Testing asset connectivity for 'carbonblack' using app 'Carbon Black Response' ",
      "status": "success"
    }
  ],
  "num_pages": 10
}
```

As a convenience, there is a `?pretty` parameter you can add that will provide additional `_pretty_*` values that are more human-readable, and give extra information so that you do not have to take the app and asset IDs and call their APIs to get the names. The following truncated example shows a single asset.

Example request

```
curl -k -u uname:pwd https://localhost/rest/app_status?pretty -G -X GET
```

Example response

```
{
  "count": 93,
  "data": [
    {
      "_pretty_app": "LDAP",
      "_pretty_asset": "domainctrl1",
      "_pretty_create_time": "13 minutes ago",
      "_pretty_icon": "/app_resource/ldap_84110F27-6602-4DC8-A6F2-0311B1720BF8/microsoft_logo.png",
      "app": 104,
      "asset": 4,
      "create_time": "2017-07-17T18:00:05.546000Z",
      "id": 1,
      "message": "Testing asset connectivity for 'domainctrl1' using app 'LDAP' ",
      "status": "success"
    },
...
  ],
  "num_pages": 10
}
```

Besides the paging and pretty parameters, the other standard query parameters and filters apply. See REST Query Data.

/rest/widget_data

Syntax

```
https://<username>:<password>@<host>/rest/widget_data
```

GET

Get data used to populate the home page widgets.

Example request

```
curl -k -u uname:pwd https://localhost/rest/widget_data -G -X GET
```

Example response

The following example is a snippet of all widget data.

```
{
  "artifact_stats": [
    [
      3,
      "Malicious URL Request Attempt",
      2,
      "100.00"
    ],
    [
      2,
      "Malicious URL Request Attempt",
      7,
      "66.67"
    ],
    [
      1,
      "Malicious URL Request Attempt",
      1,
      "33.33"
    ],
    [
      1,
      "Malicious URL Request Attempt",
      3,
      "33.33"
    ],
    [
      1,
      "Malicious URL Request Attempt",
      10,
      "33.33"
    ],
    [
      1,
      "Malicious URL Request Attempt",
      6,
      "33.33"
    ],
    [
      1,
      "Malicious URL Request Attempt",
      4,
      "33.33"
    ]
  ],
  "container_stats": {
    "groups": [
      {
        "name": "severity",
        "parts": [
          {
            "name": "high",
            "value": 3
          },
          {
            "name": "medium",
            "value": 5
          },
          {
            "name": "low",
            "value": 2
          }
        ]
      },
...
  "top_playbooks_actions": {
    "actions": [
      {
        "action_name": "url reputation",
        "average_duration": 35.6,
        "total": 7
      }
    ],
    "playbooks": []
  }
}
```

The general form of the URL to receive data for a specific widget is `/rest/widget_data/<widget-name>`. The following table shows the full list of widgets available, and optional parameters for each.

URL	PARAMETERS	DESCRIPTION
/rest/widget_data/playbook_stats	label	Returns the total number of playbooks and the number of active playbooks. The "label" parameter will cause it to return only the set of playbooks designated to operate on the supplied container label.
/rest/widget_data/sla_trend	start_time end_time label user_id	Provides averages for container resolution SLAs.

URL	PARAMETERS	DESCRIPTION
/rest/widget_data/containers_new	start_time end_time label user_id	Returns how many new containers were created per day.
/rest/widget_data/container_stats	start_time end_time label user_id	Provides various information (i.e. severity, sensitivity, closed, etc.) for returned containers.
/rest/widget_data/label_stats	start_time end_time label user_id	Provides the number of events per day.
/rest/widget_data/containers_open	label sla	Provides information (i.e. sensitivity, severity, times, owner, name, service level agreement breaches by percentage, etc.) for returned open containers.
/rest/widget_data/containers_workload	label user_id	Provides the number of open containers owned per user returned.
/rest/widget_data/pending_approvals	start_time end_time user_id	Provides information about the pending approvals per user returned.
/rest/widget_data/containers_filtered_by_status	start_time end_time label user_id	Returns priority information for containers in the sets resolved, unresolved, and all.
/rest/widget_data/top_playbooks_actions	action_filter playbook_category start_time end_time label user_id	Returns the top actions used by playbooks. The action_filter value can be "manual" or "automated". The playbook_category parameter will cause playbooks with only the given category to be returned.
/rest/widget_data/playbooks_actions_stats	start_time end_time label	Returns the number of playbooks and actions executed per day.
/rest/widget_data/containers_resolved	start_time end_time label	Returns number of containers resolved per day.

URL	PARAMETERS	DESCRIPTION
	user_id	
/rest/widget_data/roi_summary	start_time end_time label user_id	Returns ROI summary information. Note: This API returns only whole numbers dropping any decimals. Use roi_stats if you need the decimals.
/rest/widget_data/roi_stats	start_time end_time label user_id	Returns ROI and time saved information per day.
/rest/widget_data/sla_stats	start_time end_time label	Returns number of SLA breaches per day.
/rest/widget_data/containers_performance	start_time end_time label user_id	Provides averages times for triage, resolution, and dwell for the returned containers.
/rest/widget_data/artifact_stats	start_time end_time label user_id	Provides information about the top artifacts per container.

Most widgets take label and user_id as optional parameters, where the label is a container label and user_id is an integer. Widgets can also take the optional parameters start_time and end_time, which must be specified in ISO 8601 format and provided together, else they will default to the last 7 days.

Use a Custom Script

Use Splunk SOAR (On-premises) to set up your own script to handle a REST request. Setting up your own script can be useful if you have an existing tool or product that can forward to a custom URL, but can't convert its data to the native JSON structure in Splunk SOAR (On-premises). This feature is shipped as part of the Generic/REST Data Source App.

Handling REST requests

As of software version 4.9., REST handlers can't import any Splunk SOAR platform code. But, REST handlers can import any dependencies defined in your app JSON, and any of Django's built-in modules, but calling anything that depends on django.setup() fails.

Create a Python script to handle and parse the incoming REST request. The script needs to implement a function called `handle_request`. The function will take a single parameter, which is a Django Request object.

Copy and paste the following script, modify it as necessary, and save it as custom.py.

`custom.py`

```
import json

def handle_request(request):

  # For this example, the incoming request is in JSON format.
  # See the documentation link to download the sample JSON
  # to post a single container and single artifact.
  post_as_json = json.loads(request.body)


  # The example JSON has "container_data" which is the data that
  # goes into a container. Within that data, there is "artifact_data"
  # which goes into an artifact.

  result = []

  # First, gather the artifact_data
  adata = post_as_json['container_data']['artifact_data']
  cdata = post_as_json['container_data']

  # Remove artifact_data from the container data
  del cdata['artifact_data']

  container = cdata

  # You can post multiple artifacts by adding them to the list.
  # In this case, there is only one artifact in our example.
  artifact = [adata]

  # The result is a list, containing a dictionary of one container and
  # potentially multiple artifacts. You could add
  # multiple containers with multiple artifacts by appending to
  # this list.

  result.append({
    'container': container,
    'artifacts': artifact
  })

  return result
```

Copy and paste the following sample data, modify it as necessary, and save it as custom.json.

`custom.json`

```
{
  "container_data": {
    "asset_id": "4",
    "status": "new",
    "description": "7PJFMqhL1U1IuK7AuXdJKsoAea",
    "name": "qlmc1oh7nkr99ob9mpmfoq1sdm",
    "sensitivity": "amber",
    "source_data_identifier": "kmjafnviJxaXsRlaIVPm52BbAw",
    "due_time": "2016-01-14T21:54:39.517382Z",
    "end_time": "2016-01-14T21:54:39.517555Z",
    "ingest_connector_id": 437614,
"data": {
  "data_field1": "BGhXm6wizmLjt"
},
"start_time": "2016-01-14T21:54:39.517955Z",
"severity": "medium",
"artifact_data": {
  "asset_id": 601925,
  "cef": {
    "requestClientApplication": "",
    "destinationAddress": "",
    "deviceFacility": "",
    "deviceExternalId": "",
    "cn2Label": "",
    "deviceCustomDate1": "",
    "deviceCustomDate2": "",
    "cs1": "",
    "deviceCustomNumber3Label": "",
    "destinationDnsDomain": "",
    "destinationUserName": "",
    "fileHash": "8b317b683d052efda16ab03c68ae2dee",
    "filePath": "",
    "end": "",
    "sourceTranslatedAddress": "",
    "fileName": "UNAVAILABLE",
    "fileModificationTime": "",
    "deviceCustomString5Label": "",
    "deviceAddress": "",
    "destinationHostName": "",
    "fileId": "",
    "dpt": "",
    "oldfileHash": "",
    "destinationServiceName": "",
    "sourceNtDomain": "",
    "requestMethod": "",
    "cn1Label": "",
    "sourceDnsDomain": "",
    "endTime": "",
    "sourceAddress": "10.10.0.201",
    "bytesIn": "",
    "bytesOut": "",
    "deviceProcessName": "",
    "cn3Label": "",
    "deviceCustomString4Label": "",
    "cs6Label": "",
    "out": "",
    "rt": "",
    "cnt": "",
    "sourceHostName": "",
    "deviceCustomNumber3": "",
    "dst": "103.230.84.239",
    "deviceHostname": "",
    "deviceCustomNumber1Label": "",
    "externalId": "",
    "deviceDirection": "",
    "method": "",
    "dmac": "",
    "oldfileId": "",
    "sourceServiceName": "",
    "baseEventCount": "120",
    "dvchost": "",
    "destinationUserPrivileges": "",
    "fileSize": "",
"sourceMacAddress": "",
"deviceCustomString6": "",
"deviceCustomString5": "",
"deviceCustomString4": "",
"deviceCustomString3": "",
"deviceCustomString1Label": "",
"destinationMacAddress": "",
"duser": "",
"cs3Label": "",
"cn2": "",
"cn3": "",
"destinationTranslatedAddress": "",
"cn1": "",
"deviceCustomString3Label": "",
"act": "",
"dhost": "",
"filePermission": "",
"dntdom": "",
"deviceCustomString6Label": "",
"app": "",
"smac": "",
"deviceInboundInterface": "",
"message": "",
"deviceDnsDomain": "",
"deviceCustomNumber2": "",
"deviceCustomNumber1": "",
"deviceCustomDate2Label": "",
"dpriv": "",
"suid": "",
"oldfilePermission": "",
"deviceCustomString2": "",
"start": "",
"deviceCustomString1": "",
"msg": "",
"src": "",
"sourceUserId": "",
"requestURL": "",
"sourceUserName": "",
"sourceTranslatedPort": "",
"oldfileType": "",
"destinationPort": "80",
"destinationUserId": "",
"sourcePort": "4286",
"requestCookies": "",
"spriv": "",
"deviceCustomDate1Label": "",
"cs4Label": "",
"transportProtocol": "",
"cat": "",
"deviceMacAddress": "",
"duid": "",
"deviceOutboundInterface": "",
"destinationProcessName": "",
"spt": "",
"cs2Label": "",
"sntdom": "",
"in": "",
"oldfileModificationTime": "",
"oldfileName": "",
"deviceAction": "",
"proto": "",
"deviceCustomString2Label": "",
        "fileType": "",
        "deviceTranslatedAddress": "",
        "fsize": "",
        "dproc": "",
        "fname": "",
        "destinationNtDomain": "",
        "fileCreateTime": "2014-10-19 12:41:32",
        "ApplicationProtocol": "",
        "suser": "",
        "destinationTranslatedPort": "",
        "cs1Label": "",
        "oldfsize": "",
        "startTime": "09\/09\/2014 16:30:00",
        "shost": "",
        "oldfileCreateTime": "",
        "oldfilePath": "",
        "deviceCustomNumber2Label": "",
        "deviceEventCategory": "",
        "cs5Label": "",
        "request": "",
        "sourceUserPrivileges": "",
        "dvc": "",
        "cs5": "",
        "cs4": "",
        "cs6": "",
        "receiptTime": "",
        "cs3": "",
        "cs2": ""
      },
      "severity": "medium",
      "data": {
        "data_field1": "ZURNKdbqMSny8"
      },
      "start_time": "2016-01-14T21:54:39.762095Z",
      "source_data_identifier": "n4ivxGKh0tHWso74nhJlcSSwKq",
      "label": "event",
      "end_time": "2016-01-14T21:54:39.762530Z",
      "ingest_connector_id": 796153,
      "type": "NWPUAbtcNNQDyz9KXurJkdG7QV"
    }
  }
}
```

In the script example, a JSON document is expected to be POSTed. However, the data could be in XML, CSV, or any other format. The Django Request object that is passed has a `body` member that contains the content of the POST, which we convert to native Python data structures using `json.loads()`. The request content is typically all that is needed to construct your container and artifact data. The Request object contains much more than just the request content, including the HTTP method, which may be useful if you intend to handle GETs and POSTs. To find out more about Django Request objects you can visit the Django documentation at https://docs.djangoproject.com.

The resulting container and artifact objects should have the same contents as a REST POST to Splunk SOAR (On-premises), as described in the REST documentation about creating containers and artifacts. There is no need to provide the `ingest_app_id`, `asset_id`, or `label` fields, since those are already provided.

The final result from the `handle_request` function should be a list of dictionaries that contain two keys, `container`, and `artifacts`. The value for container should be a single dictionary providing the container data and the `artifacts` value is a list of dictionaries, one for each artifact to be added. The `container_ids` of the artifacts don't need to be supplied if providing a container.

Since the final return value is a list, you can create multiple containers from each REST request.

Add the following code to add logging to your script.

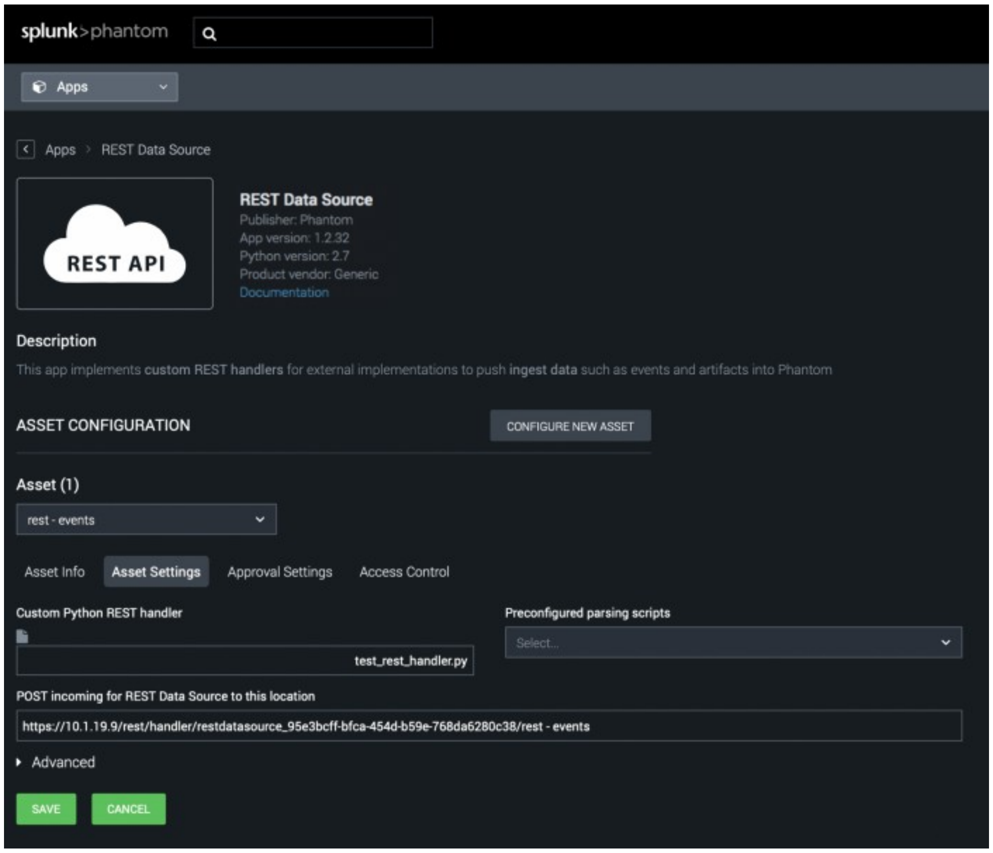
```
custom.py


import logging
logger = logging.getLogger('default')
...
# from inside your handler function
    logger.debug('this is my log message')
```

Setting your script to receive REST requests

Configure your script to receive REST requests by uploading it in the asset configuration screen.

Perform the following steps from the **Main** menu.

1. Navigate to **Apps**.
2. Scroll to REST Data Source and click **CONFIGURE NEW ASSET**.
3. Select **rest - events** from the asset drop-down.
4. Click the **Asset Settings** tab.
5. Take note of the "POST incoming for REST Data Source to this location"



REST Data Source
Publisher: Phantom
App version: 1.2.32
Python version: 2.7
Product vendor: Generic
Documentation

Description

This app implements custom REST handlers for external implementations to push ingest data such as events and artifacts into Phantom

ASSET CONFIGURATION

Asset (1)

rest - events

Custom Python REST handler: test_rest_handler.py

Preconfigured parsing scripts: Select...

POST incoming for REST Data Source to this location

https://10.1.19.9/rest/handler/restdatasource_95e3bcff-bfca-454d-b59e-768da6280c38/rest - events

Once the script has been uploaded, direct your tool or 3rd party product to post to `/rest/handler/rest_ingest/restdatasource_[AppID]/[asset name]` where `[asset name]` is the name of the asset you attached your script to. `[AppID]` is a unique App ID belonging to the installed REST App, as shown in the screenshot.

You can use the following curl command to test your parse script if you have sample data for testing, such as custom.json. `curl -d "@./custom.json" --insecure -H "ph-auth-source:asset_name" -H "ph-auth-token:..." https://x.x.x.x/rest/handler/rest_ingest//restdatasource_[AppID]/asset_name`

The artifact dictionary does not set the 'container_id' value in this case because the container and artifacts are returned together by the custom script to the platform.

When using the REST Handler app from Splunk Phantom version 4.9 on, you must use the REST Handler app version 1.2.36 or newer.

Configured scripts

Splunk SOAR (On-premises) provides scripts to handle data in standard formats or from some popular products. Select these scripts in Asset Settings.

Perform the following steps from the **Main** menu.

1. Navigate to **Apps**.
2. Scroll to REST Data Source and click **CONFIGURE NEW ASSET**.
3. Select '*rest - events* from the asset menu.
4. Click the **Asset Settings** tab.
5. Select a script from the **Preconfigured parsing scripts** menu.

Splunk SOAR (On-premises) provides the following scripts that you can modify to work in your environment. You can upload your modified copy as a custom script.

FireEye

fireeye_rest_handler.py

```
#!/bin/env python
# --
# File: fireeye_rest_handler.py
# Copyright (c) 2016-2020 Splunk Inc.
#
# SPLUNK CONFIDENTIAL - Use or disclosure of this material in whole or in part
# without a valid written license from Splunk Inc. is PROHIBITED.
from six import string_types
import sys
import json
import email
from parse import parse

ARTIFACT_LABEL_ALERT = "Alert"
ARTIFACT_LABEL_ANALYSIS = "Analysis"

# dictionary that contains the comman keys in the container
_container_common = {
    "description": "Container added by Phantom",
    "run_automation": False  # Don't run any playbooks, when this container is added
}
_artifact_common = {
    "type": "network",
    "description": "Artifact added by Phantom",
    "run_automation": False  # Don't run any playbooks, when this artifact is added
}


def _get_value(in_dict, in_key, def_val=None, strip_it=True):
    if in_key not in in_dict:
        return def_val

    if not isinstance(in_dict[in_key], string_types):
        return in_dict[in_key]

    value = in_dict[in_key].strip() if strip_it else in_dict[in_key]

    return value if len(value) else def_val


def _set_cef_key(src_dict, src_key, dst_dict, dst_key):
    src_value = _get_value(src_dict, src_key)

    # if None, try again after removing the @ char
    if src_value is None:
        if src_key.startswith('@'):
            return _set_cef_key(src_dict, src_key[1:], dst_dict, dst_key)
        return False

    dst_dict[dst_key] = src_value

    return True


def set_url(http_header, cef):
    # get the request line
    request, header_str = http_header.split('\r\n', 1)
    headers = email.message_from_string(header_str)

    # Remove multiple spaces if any. always happens in http request line
    request = ' '.join(request.split())

    url = request.split()[1]

    if url:
        host = headers.get('Host')
        if url.startswith('http') is False:
            if host:
                url = 'http://{0}{1}'.format(host, url)

        cef['requestURL'] = url

    return


def parse_time(input_time):
    # format to match "2013-03-28 22:41:39+00"
    result = parse("{year}-{month}-{day} {hour}:{min}:{secs}+00", input_time)
    if result is not None:
        return "{year}-{month}-{day}T{hour}:{min}:{secs}.0Z".format(**result.named)
    # format to match "2013-03-28T22:41:39Z"
    result = parse("{year}-{month}-{day}T{hour}:{min}:{secs}Z", input_time)
    if result is not None:
        return "{year}-{month}-{day}T{hour}:{min}:{secs}.0Z".format(**result.named)

    # Return the input as is, if the rest endpoint does not like it, it will return an error
    return input_time


def parse_alert(alert, result):
    new_data = {}
    result.append(new_data)

    # Create the container, each alert represents a container
    container = dict()
    new_data['container'] = container
    container.update(_container_common)
    container['name'] = alert.get('@name', alert.get('name'))

    if '@id' not in alert:
        if 'id' not in alert:
            raise TypeError('id key not found in alert')

    container['source_data_identifier'] = alert.get('@id', alert.get('id'))
    container['data'] = alert

    start_time = alert.get('occurred')
    if start_time:
        start_time = parse_time(start_time)
        container['start_time'] = start_time

    severity = alert.get('@severity', alert.get('severity', 'medium'))
    container['severity'] = 'high' if severity == 'crit' else 'medium'

    artifact_label = ARTIFACT_LABEL_ALERT

    if container['name'] == 'malware-object':
        artifact_label = ARTIFACT_LABEL_ANALYSIS

    # now the artifacts
    new_data['artifacts'] = artifacts = []

    artifact = dict()
    artifacts.append(artifact)

    artifact.update(_container_common)
    artifact.update(_artifact_common)
    artifact['label'] = artifact_label
    artifact_id = len(artifacts)
    artifact['name'] = "Artifact ID: {0}".format(artifact_id)
    artifact['source_data_identifier'] = str(artifact_id)

    start_time = alert.get('occurred')
    if start_time:
        start_time = parse_time(start_time)
        container['start_time'] = start_time

    artifact['cef'] = cef = dict()

    dst = alert.get('dst')
    if dst:
        _set_cef_key(dst, 'host', cef, 'destinationHostName')
        _set_cef_key(dst, 'ip', cef, 'destinationAddress')
        _set_cef_key(dst, 'port', cef, 'destinationPort')
        _set_cef_key(dst, 'mac', cef, 'destinationMacAddress')

    src = alert.get('src')
    if src:
        _set_cef_key(src, 'host', cef, 'sourceHostName')
        _set_cef_key(src, 'ip', cef, 'sourceAddress')
        _set_cef_key(src, 'port', cef, 'sourcePort')
        _set_cef_key(src, 'mac', cef, 'sourceMacAddress')

    intf = alert.get('interface')
    if intf:
        _set_cef_key(intf, 'interface', cef, 'deviceInboundInterface')

    explanation = alert.get('explanation')
    if explanation:
        _set_cef_key(explanation, '@protocol', cef, 'transportProtocol')

    # Artifact for malware-detected
    mal_detected = explanation.get('malware-detected')
    if mal_detected:
        malware = mal_detected.get('malware')
        if malware:
            artifact = dict()
            artifacts.append(artifact)
            artifact.update(_container_common)
            artifact.update(_artifact_common)
            artifact['label'] = artifact_label
            artifact['source_data_identifier'] = len(artifacts)
            artifact['name'] = "Malware Detected "
            artifact['cef'] = cef = dict()
            cef['cs1Label'] = 'signatureName'
            _set_cef_key(malware, '@name', cef, 'cs1')
            cef['cs2Label'] = 'signatureId'
            _set_cef_key(malware, '@sid', cef, 'cs2')
            _set_cef_key(malware, 'application', cef, 'fileName')
            _set_cef_key(malware, 'original', cef, 'filePath')
            _set_cef_key(malware, 'md5sum', cef, 'fileHash')
            _set_cef_key(malware, 'downloaded-at', cef, 'fileCreateTime')
            cef['cs3Label'] = 'httpHeader'
            _set_cef_key(malware, 'http-header', cef, 'cs3')
            if 'http-header' in malware:
                set_url(cef['cs3'], cef)

    # Artifact for cnc-services
    cnc_services = explanation.get('cnc-services')
    if cnc_services:
        cnc_service = cnc_services.get('cnc-service')
        if cnc_service:
            if type(cnc_service) == dict:
                cnc_services_list = []
                cnc_services_list.append(cnc_service)
                cnc_service = cnc_services_list

            for i, service in enumerate(cnc_service):
                artifact = dict()
                artifacts.append(artifact)
                artifact.update(_container_common)
                artifact.update(_artifact_common)
                artifact['label'] = artifact_label
                artifact['source_data_identifier'] = len(artifacts)
                artifact['name'] = "CNC Service # {0}".format(i)
                artifact['cef'] = cef = dict()
                _set_cef_key(service, '@port', cef, 'destinationPort')
                _set_cef_key(service, '@protocol', cef, 'transportProtocol')
                _set_cef_key(service, 'address', cef, 'destinationAddress')
                cef['deviceDirection'] = 'out'
                cef['cs1Label'] = 'channel'
                _set_cef_key(service, 'channel', cef, 'cs1')
                if 'channel' in service:
                    sanitized_header = cef['cs1'].replace('::~~', '\r\n')
                    try:
                        set_url(sanitized_header, cef)
                    except:
                        # Most probably, not a valid http header
                        pass

    return


def parse_json(input_json):
    result = []

    try:
        fe_json = json.loads(input_json)
    except Exception as e:
        return "Unable to parse input json file, possibly incorrect format. Parse Error:
{0}".format(e.message)

    alerts = fe_json.get('alert')

    source_device_name = fe_json.get('@appliance', fe_json.get('appliance', ''))

    _artifact_common['deviceHostname'] = source_device_name

    if type(alerts) == dict:
        alerts_list = []
        alerts_list.append(alerts)
        alerts = alerts_list

    for alert in alerts:
        parse_alert(alert, result)

    return result


def handle_request(request):
    return parse_json(request.body)


if __name__ == '__main__':

    with open(sys.argv[1]) as f:
        result = parse_json(str(f.read()))
        # import pprint;pprint.pprint(result)
        print(json.dumps(result))

    exit(0)
```

STIX

stix_rest_handler.py

```
#!/usr/bin/env python
# File: stix_rest_handler.py
# Copyright (c) 2016-2020 Splunk Inc.
#
# SPLUNK CONFIDENTIAL - Use or disclosure of this material in whole or in part
# without a valid written license from Splunk Inc. is PROHIBITED.

import libtaxii as lt
from stix.core import STIXPackage
from collections import OrderedDict
from jsonpath_rw import parse as jp_parse
from six import string_types
import uuid
from copy import deepcopy
import json
from phantom_common.compat import StringIO

# dictionary that contains the common keys in the container
_container_common = {
    "description": "Container added by Splunk SOAR (On-premises)",
    "run_automation": False  # Don't run any playbooks, when this container is added
}
_artifact_common = {
    "type": "network",
    "description": "Artifact added by Splunk SOAR (On-premises)",
    "run_automation": False  # Don't run any playbooks, when this artifact is added
}


def process_results(results):
    processed_results = []

    for i, result in enumerate(results):

        # container is a dictionary of a single container and artifacts
        if 'container' not in result:
            continue

        # container is a dictionary of a single container and artifacts
        if not result.get('artifacts'):
            # igonore containers without artifacts
            continue

        for j, artifact in enumerate(result['artifacts']):

            if 'source_data_identifier' not in artifact:
                artifact['source_data_identifier'] = j

            artifact.update(_artifact_common)

        processed_results.append(result)

    return processed_results


def handle_request(f):
    # The django request object does not support seek, so move it to cSTringIO
    cstrio = StringIO()
    cstrio.write(f.read().decode('utf-8'))
    cstrio.seek(0)

    # first try to parse it as a taxii message
    try:
        taxii_msg = lt.tm11.get_message_from_xml(cstrio.read())
    except:
        # Now as a a stix document
        cstrio.seek(0)
        package = parse_stix(cstrio)

        if type(package) == str:
            # Error
            return package

        packages = [package]
        results = parse_packages(packages, None)
    else:
        results = parse_taxii_message(taxii_msg, None)

    # import pprint;pprint.pprint(results)
    # with open('/tmp/taxii-parsed.json', 'w') as f:
    #     f.write(json.dumps(results, indent=' ' * 4))

    return process_results(results)

# -------- Stix -------


get_list = lambda x: x if type(x) is list else [x]


def parse_domain_obj_type(prop, obs_json):
    if 'value' not in prop:
        return

    value = prop['value']

    if isinstance(value, string_types):
        cef = dict()
        artifact = dict()
        _set_cef_key(prop, 'value', cef, 'destinationDnsDomain')

        # so this artifact needs to be added
        artifact['name'] = "Domain Object"
        artifact['cef'] = cef
        # append to the properties
        obs_json['properties'].append(artifact)
    elif type(value) == dict:
        if 'value' in value:
            value = value['value']
            value = get_list(value)  # convert to list, removes requirement for if else
            for addr in value:
                cef = dict()
                artifact = dict()
                cef['destinationDnsDomain'] = addr
                # so this artifact needs to be added
                artifact['name'] = "Domain Object"
                artifact['cef'] = cef
                # append to the properties
                obs_json['properties'].append(artifact)

    return


def parse_hash_object(file_hash, obs_json, file_name=None, file_size=None, file_path=None):
    if file_hash is None:
        return

    if 'simple_hash_value' not in file_hash:
        return

    hash_value = file_hash['simple_hash_value']

    ret_val = False

    if isinstance(hash_value, string_types):
        cef = dict()
        _set_cef_key(file_hash, 'simple_hash_value', cef, 'fileHash')
        if len(cef) == 0:
            return
        if file_name:
            cef['fileName'] = file_name
        if file_size:
            cef['fileSize'] = file_size
        if file_path:
            cef['filePath'] = file_path
        artifact = dict()
        # so this artifact needs to be added
        artifact['name'] = "File Object"
        artifact['cef'] = cef
        # append to the properties
        obs_json['properties'].append(artifact)
        return True
    elif type(hash_value) == dict:
        if 'value' in hash_value:
            value = hash_value['value']
            value = get_list(value)  # convert to list, removes requirement for if else
            for curr_hash in value:
                cef = dict()
                artifact = dict()
                cef['fileHash'] = curr_hash
                if file_name:
                    cef['fileName'] = file_name
                if file_size:
                    cef['fileSize'] = file_size
                if file_path:
                    cef['filePath'] = file_path
                # so this artifact needs to be added
                artifact['name'] = "File Object"
                artifact['cef'] = cef
                # append to the properties
                obs_json['properties'].append(artifact)
                ret_val = True

    return ret_val


def parse_file_name_obj(file_name, prop):
    if type(file_name) == dict:
        value = file_name.get('value')
        if not value:
            return None

        condition = file_name.get('condition')
        if condition.lower() == 'contains':
            return "*{0}*".format(value)
        elif condition.lower() == 'equals':
            return value

    return None


def parse_file_path_obj(file_path, prop):
    if type(file_path) == dict:
        value = file_path.get('value')
        if not value:
            return None

        condition = file_path.get('condition')
        if condition.lower() == 'contains':
            return "*{0}*".format(value)
        elif condition.lower() == 'equals':
            return value

    return None


def parse_email_address(email_object, email_type, prop, obs_json):
    if not email_object:
        return

    cef_key = 'cs1' if (email_type == 'from') else 'cs2'
    cef_label = 'fromEmail' if (email_type == 'from') else 'toEmail'

    category = email_object.get('category')

    if not category:
        return

    if (category != 'e-mail') and (category != 'email'):
        return

    if email_object.get('xsi:type') != 'AddressObjectType':
        return

    email_addr_value = email_object.get('address_value')
    if not email_addr_value:
        return

    artifacts = parse_common_obj_type(email_addr_value, obs_json, 'value', cef_key, 'Email Object')
    try:
        for artifact in artifacts:
            artifact['cef']['cs1Label'] = cef_label
    except:
        pass

    return


def parse_email_obj_type(prop, obs_json):
    header = prop.get('header')

    if not header:
        return

    from_object = header.get('from')

    if from_object:
        parse_email_address(from_object, 'from', prop, obs_json)

    to_object = header.get('to')

    if to_object:
        parse_email_address(to_object, 'to', prop, obs_json)

    subject_object = header.get('subject')

    if subject_object:
        artifacts = parse_common_obj_type(subject_object, obs_json, 'value', 'cs3', 'Email Object')
        try:
            for artifact in artifacts:
                artifact['cef']['cs3Label'] = 'subject'
        except:
            pass

    return


def parse_win_reg_key_obj_type(prop, obs_json):
    hive = prop.get('hive')

    if not hive:
        return

    cef = {}
    _set_cef_key(hive, 'value', cef, 'cs1', 'cs1Label', 'hive')

    key = prop.get('key')
    if key:
        _set_cef_key(key, 'value', cef, 'cs2', 'cs2Label', 'key')

    values = prop.get('values')

    for value in values:

        if not value:
            continue

        name = value.get('name')

        if not name:
            continue

        data = value.get('data')

        if not data:
            continue

        curr_cef = dict(cef)

        _set_cef_key(name, 'value', curr_cef, 'cs3', 'cs3Label', 'name')
        _set_cef_key(data, 'value', curr_cef, 'cs4', 'cs4Label', 'data')

        artifact = dict()
        artifact['name'] = "Registry Object"
        artifact['cef'] = curr_cef
        obs_json['properties'].append(artifact)

    return


def parse_file_obj_type(prop, obs_json):
    # Check if hashes are present
    hashes = prop.get('hashes')
    file_name = prop.get('file_name')
    file_size = prop.get('size_in_bytes')
    file_path = prop.get('file_path')

    if file_path:
        file_path = parse_file_path_obj(file_path, prop)

    if file_name:
        file_name = parse_file_name_obj(file_name, prop)

    hash_added = 0

    if hashes:
        for curr_hash in hashes:
            hash_added |= parse_hash_object(curr_hash, obs_json, file_name, file_size, file_path)

        if hash_added:
            # FileHash added, no need to add anymore properties
            return

    # if hashes could not be added for some reason (or not present), need
    # to add the file name and size if available as an artifact on it's own
    cef = dict()
    _set_cef_key(prop, 'file_name', cef, 'fileName')
    _set_cef_key(prop, 'size_in_bytes', cef, 'fileSize')
    if file_path:
        cef['filePath'] = file_path
    if len(cef) == 0:
        return
    artifact = dict()
    artifact['name'] = "File Object"
    artifact['cef'] = cef
    obs_json['properties'].append(artifact)

    return


def parse_port_obj_type(prop, obs_json):
    # first store the protocol value
    protocol_value = prop.get('layer4_protocol')

    if protocol_value is None:
        # keep it empty
        protocol_value = dict()

    port_value = prop.get('port_value')

    if port_value is None:
        return

    if isinstance(port_value, string_types):
        artifact = dict()
        cef = dict()
        _set_cef_key(protocol_value, 'value', cef, 'transportProtocol')
        _set_cef_key(prop, 'port_value', cef, 'destinationPort')
        # so this artifact needs to be added
        artifact['name'] = "Port Object"
        artifact['cef'] = cef
        # append to the properties
        obs_json['properties'].append(artifact)

    elif type(port_value) == dict:
        condition = port_value.get('condition')
        if condition is None:
            return

        value = port_value.get('value')
        if condition == 'InclusiveBetween':
            artifact = dict()
            cef = dict()
            _set_cef_key(protocol_value, 'value', cef, 'transportProtocol')
            cef['destinationPort'] = '-'.join(value)
            artifact['name'] = "Port Object"
            artifact['cef'] = cef
            # append to the properties
            obs_json['properties'].append(artifact)

        elif condition == 'Equals':
            value = get_list(value)  # convert to list, removes requirement for if else
            for addr in value:
                artifact = dict()
                cef = dict()
                _set_cef_key(protocol_value, 'value', cef, 'transportProtocol')
                cef['destinationPort'] = addr
                artifact['name'] = "Port Object"
                artifact['cef'] = cef
                # append to the properties
                obs_json['properties'].append(artifact)

    return


def parse_address_obj_type(prop, obs_json):
    addr_value = prop.get('address_value')

    if addr_value is None:
        return

    if isinstance(addr_value, string_types):
        artifact = dict()
        cef = dict()
        _set_cef_key(prop, 'address_value', cef, 'destinationAddress')
        # so this artifact needs to be added
        artifact['name'] = "Address Object"
        artifact['cef'] = cef
        # append to the properties
        obs_json['properties'].append(artifact)

    elif type(addr_value) == dict:
        condition = addr_value.get('condition')
        if condition is None:
            return

        value = addr_value.get('value')
        if condition == 'InclusiveBetween':
            artifact = dict()
            cef = dict()
            cef['destinationAddress'] = '-'.join(value)
            artifact['name'] = "Address Object"
            artifact['cef'] = cef
            # append to the properties
            obs_json['properties'].append(artifact)
        elif condition == 'Equals':
            value = get_list(value)  # convert to list, removes requirement for if else
            for addr in value:
                artifact = dict()
                cef = dict()
                cef['destinationAddress'] = addr
                artifact['name'] = "Address Object"
                artifact['cef'] = cef
                # append to the properties
                obs_json['properties'].append(artifact)


def parse_common_obj_type(prop, obs_json, key_name, cef_key, artifact_name):
    addr_value = prop.get(key_name)

    if addr_value is None:
        return None

    artifacts = []

    if isinstance(addr_value, string_types):
        artifact = dict()
        cef = dict()
        _set_cef_key(prop, key_name, cef, cef_key)
        # so this artifact needs to be added
        artifact['name'] = artifact_name
        artifact['cef'] = cef
        # append to the properties
        artifacts.append(artifact)
        obs_json['properties'].append(artifact)

    elif type(addr_value) == dict:
        condition = addr_value.get('condition')
        if condition is None:
            return None

        value = addr_value.get('value')
        if condition == 'InclusiveBetween':
            artifact = dict()
            cef = dict()
            cef[cef_key] = '-'.join(value)
            artifact['name'] = artifact_name
            artifact['cef'] = cef
            # append to the properties
            artifacts.append(artifacts)
            obs_json['properties'].append(artifact)

        elif condition == 'Equals':
            value = get_list(value)  # convert to list, removes requirement for if else
            for addr in value:
                artifact = dict()
                cef = dict()
                cef[cef_key] = addr
                artifact['name'] = artifact_name
                artifact['cef'] = cef
            # append to the properties
            artifacts.append(artifacts)
            obs_json['properties'].append(artifact)

    return artifacts


def parse_uri_obj_type(prop, obs_json):
    uri_type = prop.get('type')
    if uri_type is None:
        parse_common_obj_type(prop, obs_json, 'value', 'requestURL', 'URI Object')
    elif uri_type == 'Domain Name':
        parse_common_obj_type(prop, obs_json, 'value', 'destinationDnsDomain', 'Domain Object')
    elif uri_type == 'URL':
        parse_common_obj_type(prop, obs_json, 'value', 'requestURL', 'URI Object')

    return


def parse_sock_obj_type(prop, obs_json):
    ip_addr = prop.get('ip_address')

    if ip_addr:
        if ip_addr['xsi:type'] == 'AddressObjectType':
            parse_address_obj_type(ip_addr, obs_json)

    return


def parse_net_conn_obj_type(prop, obs_json):
    dest_sock_addr = prop.get('destination_socket_address')

    if dest_sock_addr:
        if dest_sock_addr['xsi:type'] == 'SocketAddressObjectType':
            parse_sock_obj_type(dest_sock_addr, obs_json)

    return


def parse_property(prop, obs_json):
    try:
        if prop['xsi:type'] == 'FileObjectType':
            parse_file_obj_type(prop, obs_json)
        elif prop['xsi:type'] == 'DomainNameObjectType':
            parse_domain_obj_type(prop, obs_json)
        elif prop['xsi:type'] == 'AddressObjectType':
            parse_address_obj_type(prop, obs_json)
        elif prop['xsi:type'] == 'URIObjectType':
            parse_uri_obj_type(prop, obs_json)
        elif prop['xsi:type'] == 'PortObjectType':
            parse_port_obj_type(prop, obs_json)
        elif prop['xsi:type'] == 'NetworkConnectionObjectType':
            parse_net_conn_obj_type(prop, obs_json)
        elif prop['xsi:type'] == 'EmailMessageObjectType':
            parse_email_obj_type(prop, obs_json)
        elif prop['xsi:type'] == 'WindowsRegistryKeyObjectType':
            parse_win_reg_key_obj_type(prop, obs_json)
    except:
        pass

    return
```

```python
package_base = OrderedDict({
        'id': '',
        'idref': '',
        'campaigns': None,
        'coas': None,
        'exploit_targets': None,
        'incidents': None,
        'indicators': None,
        'observables': None,
        'related_packages': None,
        'reports': None,
        'threat_actors': None,
        'timestamp': '',
        'ttps': None,
        'version': ''})


def parse_indicator(indicator, package):
    # First check if it is an idref
    idref = indicator.idref
    if idref:
        # An idref means this indicator is defined elsewhere in the document
        # and the parsing code _will_ parse that incident and add artifacts anyways
        # so no need to parse this object
        return None

    jp_expr = jp_parse("$..observable")

    indicator_json = {}
    indicator_id = indicator.id_

    if not indicator_id:
        indicator_id = "Phantom:Indicator-{0}".format(uuid.uuid4())

    package['indicators'][indicator_id] = indicator_json
    indicator_dict = indicator.to_dict()

    # with open("indicator.json", "w") as f:
    #     f.write()

    indicator_json['title'] = indicator_dict.get('title')
    indicator_json['description'] = indicator_dict.get('description')

    indicator_json['input_data'] = json.dumps(indicator.to_json())

    matches = jp_expr.find(indicator_dict)

    if not matches:
        return indicator_id

    indicator_json['observable_idrefs'] = []

    for match in matches:
        observable = match.value
        if 'idref' in observable:
            indicator_json['observable_idrefs'].append(observable['idref'])
            continue

        obs_id = parse_observable(observable, package)
        if obs_id:
            indicator_json['observable_idrefs'].append(obs_id)
    return indicator_id


def parse_construct(construct, name, package):
    jp_expr = jp_parse("$..observables")

    construct_json = {}
    construct_id = construct.id_

    if not construct_id:
        construct_id = "Phantom:{0}-{1}".format(name, uuid.uuid4())

    package['{0}s'.format(name)][construct_id] = construct_json
    construct_dict = construct.to_dict()

    construct_json['title'] = construct_dict.get('title')
    construct_json['description'] = construct_dict.get('description')

    construct_json['input_data'] = json.dumps(construct.to_json())

    matches = jp_expr.find(construct.to_dict())

    if not matches:
        return construct_id

    construct_json['observable_idrefs'] = []

    for match in matches:
        observables = match.value
        for observable in observables:
            if 'idref' in observable:
                construct_json['observable_idrefs'].append(observable['idref'])
                continue

            obs_id = parse_observable(observable, package)
            if obs_id:
                construct_json['observable_idrefs'].append(obs_id)

    return construct_id


def parse_ttp(ttp, package):
    jp_expr = jp_parse("$..observables")

    ttp_json = {}
    ttp_id = ttp.id_

    if not ttp_id:
        ttp_id = "Phantom:ttp-{0}".format(uuid.uuid4())

    package['ttps'][ttp_id] = ttp_json
    ttp_dict = ttp.to_dict()

    ttp_json['title'] = ttp_dict.get('title')
    ttp_json['description'] = ttp_dict.get('description')

    ttp_json['input_data'] = json.dumps(ttp.to_json())

    matches = jp_expr.find(ttp.to_dict())

    if not matches:
        return ttp_id

    ttp_json['observable_idrefs'] = []

    for match in matches:
        observables = match.value
        for observable in observables:
            if 'idref' in observable:
                ttp_json['observable_idrefs'].append(observable['idref'])
                continue

            obs_id = parse_observable(observable, package)
            if obs_id:
                ttp_json['observable_idrefs'].append(obs_id)

    return ttp_id


def parse_report_observables(report, package):
    report_json = {}
    report_id = report.id_

    if not report_id:
        report_id = "Phantom:report-{0}".format(uuid.uuid4())

    package['reports'][report_id] = report_json
    report_dict = report.to_dict()

    report_json['title'] = report_dict.get('title')
    report_json['description'] = report_dict.get('description')

    report_json['input_data'] = json.dumps(report.to_json())

    report_dict = report.to_dict()

    if 'observables' not in report_dict:
        return report_id

    if 'observables' not in report_dict['observables']:
        return report_id

    report_json['observable_idrefs'] = []

    for observable in report_dict['observables']['observables']:
        if 'idref' in observable:
            report_json['observable_idrefs'].append(observable['idref'])
            continue

        obs_id = parse_observable(observable, package)
        if obs_id:
            report_json['observable_idrefs'].append(obs_id)

    return report_id


def parse_observable(observable, package):
    obs_json = {}
    obs_json['observable_idrefs'] = []

    obs_id = observable.get('id')
    if not obs_id:
        obs_id = "Phantom:Observable-{0}".format(uuid.uuid4())
    package['observables'][obs_id] = obs_json

    # Parse any observables in this observable
    jp_expr = jp_parse("$..observables")

    matches = jp_expr.find(observable)

    if matches:
        for match in matches:
            obs_comps = match.value
            for obs_comp in obs_comps:
                if 'idref' in obs_comp:
                    obs_json['observable_idrefs'].append(obs_comp['idref'])
                    continue

                obs_comp_id = parse_observable(obs_comp, package)
                if obs_comp_id:
                    obs_json['observable_idrefs'].append(obs_comp_id)

    # Parse any object references in this observable
    jp_expr = jp_parse("$..object_reference")

    matches = jp_expr.find(observable)

    if matches:
        for match in matches:
            obj_ref_id = match.value
            obs_json['observable_idrefs'].append(obj_ref_id)
            continue

    # Parse the properties
    jp_expr = jp_parse("$..properties")
    matches = jp_expr.find(observable)

    if not matches:
        return obs_id

    # Parse the properties
    obs_json['properties'] = []
    for match in matches:
        parse_property(match.value, obs_json)

    return obs_id


def parse_report(report, package):
    # Indicators
    if report.indicators:
        if not package.get('indicators'):
            package['indicators'] = OrderedDict()
        for inc in report.indicators:
            parse_indicator(inc, package)

    # Observable
    if report.observables:
        if not package.get('reports'):
            package['reports'] = OrderedDict()
        parse_report_observables(report, package)

    return
```

```
def parse_stix(xml_file_object, base_connector=None):
    if xml_file_object is None:
        if base_connector:
            base_connector.debug_print("Invalid input xml_file_object")
        return None

    try:
        stix_pkg = STIXPackage.from_xml(xml_file_object)
    except Exception as e:
        message = "Possibly invalid stix or taxii xml. Error: {0}".format(e.message)
        if base_connector:
            base_connector.debug_print(message)
        return message

    package = OrderedDict(package_base)
    package['id'] = stix_pkg.id_
    if not package['id']:
        package['id'] = "Phantom:Package-{0}".format(uuid.uuid4())

    package['idref'] = stix_pkg.idref
    package['version'] = stix_pkg.version
    package['timestamp'] = str(stix_pkg.timestamp)
    package['observable_idrefs'] = []
    package['observables'] = OrderedDict()
    package['input_data'] = json.dumps(stix_pkg.to_json())

    # Indicators
    if stix_pkg.indicators:
        package['indicators'] = OrderedDict()
        for inc in stix_pkg.indicators:
            parse_indicator(inc, package)

    # TTPs
    if stix_pkg.ttps:
        package['ttps'] = OrderedDict()
        for ttp in stix_pkg.ttps:
            parse_ttp(ttp, package)

    # Reports
    if stix_pkg.reports:
        for report in stix_pkg.reports:
            parse_report(report, package)

    # Observable
    if stix_pkg.observables:
        for observable in stix_pkg.observables:
            if observable.idref:
                package['observable_idrefs'].append(observable.idref)
                continue
            parse_observable(observable.to_dict(), package)

    return package


def _get_value(in_dict, in_key, def_val=None, strip_it=True):
    if in_key not in in_dict:
        return def_val

    if not isinstance(in_dict[in_key], string_types):
        return in_dict[in_key]
    value = in_dict[in_key].strip() if strip_it else in_dict[in_key]

    return value if len(value) else def_val


def _set_cef_key(src_dict, src_key, dst_dict, dst_key, cs_label_key=None, cs_label_value=None):
    src_value = _get_value(src_dict, src_key)

    # Ignore if None
    if src_value is None:
        return False

    dst_dict[dst_key] = src_value

    if cs_label_key:
        dst_dict[cs_label_key] = cs_label_value

    return True


def get_artifacts_from_observable(obs_id, observables, label):
    observable_artifacts = []
    observable = observables.get(obs_id)
    if not observable:
        # We were given an idref that points to an observable that was not defined.
        return observable_artifacts

    if 'observable_idrefs' in observable:
        for observable_idref in observable['observable_idrefs']:
            obs_artifacts = get_artifacts_from_observable(observable_idref, observables, label)
            observable_artifacts.extend(obs_artifacts)

    if 'properties' in observable:
        observable_artifacts.extend(deepcopy(observable['properties']))

    if observable_artifacts:
        for observable_artifact in observable_artifacts:
            observable_artifact['label'] = label
            observable_artifact['source_data_identifier'] = obs_id

    return observable_artifacts


def create_artifacts_from_construct(package, name, observables, artifacts):
    if not package:
        return

    constructs = package.get(name)

    if not constructs:
        return

    for construct in constructs:

        construct_artifacts = []
        curr_construct = constructs[construct]
        if 'observable_idrefs' not in curr_construct:
            continue

        for observable_idref in curr_construct['observable_idrefs']:
            obs_artifacts = get_artifacts_from_observable(observable_idref, observables, name)
            construct_artifacts.extend(obs_artifacts)
        if construct_artifacts:
            # container = {}
            # if (not curr_construct.get('title')):
            #     container['name'] = construct
            # else:
            #     container['name'] = curr_construct['title']
            # container['source_data_identifier'] = construct
            # container['data'] = curr_construct['input_data']
            artifacts.extend(construct_artifacts)

    return


def create_container_from_package(package, observables, base_connector):
    if not package:
        return {}

    artifacts = []

    constructs = ['campaigns', 'coas', 'exploit_targets', 'threat_actors', 'related_packages', 'indicators',
'reports', 'ttps']

    for construct in constructs:
        create_artifacts_from_construct(package, construct, observables, artifacts)

    if not artifacts:
        # The package probably did not contain anything but observables in the package node
        if not observables:
            # Return empty container
            return {}

        # Create a container from the package itself
        if 'observable_idrefs' in package:
            for observable_idref in package['observable_idrefs']:
                obs_arts = get_artifacts_from_observable(observable_idref, observables, 'observable')
                artifacts.extend(obs_arts)

        if 'observables' in package:
            for observable_idref in package['observables']:
                obs_arts = get_artifacts_from_observable(observable_idref, observables, 'observable')
                artifacts.extend(obs_arts)

    if artifacts:
        container = {}
        container['name'] = package['id']
        container['source_data_identifier'] = package['id']
        container['data'] = package['input_data']
        return {'container': container, 'artifacts': artifacts}

    # Return empty container
    return {}


def parse_packages(packages, base_connector):
    containers = []

    if not packages:
        if base_connector:
            base_connector.save_progress("Zero packages found")
        return containers
    # get all the observables
    if base_connector:
        base_connector.send_progress("Extracting Observables")

    jp_expr = jp_parse("$..observables")

    all_observables = OrderedDict()

    matches = jp_expr.find(packages)
    for match in matches:
        try:
            all_observables.update(match.value)
        except:
            raise

    if base_connector:
        base_connector.send_progress(" ")

    if base_connector:
        base_connector.save_progress("Creating Containers and Artifacts from {0}
packages".format(len(packages)))

    # Now look at each of the package
    for j, package in enumerate(packages):
        if base_connector:
            base_connector.send_progress("Working on STIX Package # {0}".format(j))
        package_containers = create_container_from_package(package, all_observables, base_connector)
        if package_containers:
            containers.append(package_containers)

    if base_connector:
        base_connector.send_progress(" ")

    return containers
# -------- Stix -------

# -------- Taxii ------


def parse_taxii_message(taxii_message, base_connector=None):
    number_of_cbs = len(taxii_message.content_blocks)

    if not number_of_cbs:
        return {'error': 'no control blocks found'}

    packages = []

    for i, cb in enumerate(taxii_message.content_blocks):

        if base_connector:
            base_connector.send_progress("Parsing Content Block # {0}".format(i))

        # Give it to the stix parser to create the containers and artifacts
        # This code is the only place where the stix parsing will be written
        stix_xml = cb.content
        cstrio = StringIO()
        cstrio.write(stix_xml)
        cstrio.seek(0)

        package = parse_stix(cstrio, base_connector)

        if package:
            # print (json.dumps(package, indent=' ' * 4))
            packages.append(package)

    return parse_packages(packages, base_connector)

# -------- Taxii ------
```

Administration endpoints

REST administration

/rest/indicator_cef_filter

`/rest/indicator_cef_filter`
List all `indicator_cef_filter` records.

GET

List all `indicator_cef_filter` records.

Response values

Field	Required	Type	Description
cef_type		string	Whether or not the CEF record is created by Splunk SOAR or the customer. The possible CEF types are `default` or `custom`.
cef		number	The ID of the associated CEF record.
cef_name		string	The name of the associated CEF record.
apply_filter		Boolean	Returns `true` if the associated CEF record will be filtered out during indicator creation.

JSON response

```
<div>
{
    "count": 155,
    "data": [
        {
            "cef_name": "dmac",
            "cef": 1,
            "cef_type": "default",
            "id": 1,
            "apply_filter": false
        },
        {
            "cef_name": "act",
            "cef": 2,
            "cef_type": "default",
            "id": 2,
            "apply_filter": false
        }
],
    "num_pages": 16
}
```

/rest/indicator_cef_filter/[ID]

`/rest/indicator_cef_filter/[ID]`
Get a particular `indicator_cef_filter` record by ID.

GET

Get a particular `indicator_cef_filter` record by ID.

Response values

Field	Required	Type	Description
cef_type		string	Whether or not the CEF record is created by Splunk SOAR or the customer. The possible CEF types are `default` or `custom`.
cef		number	The ID of the associated CEF record.
cef_name		string	The name of the associated CEF record.
apply_filter		boolean	Returns `true` if the associated CEF record will be filtered out during indicator creation.

JSON response

```
<div>
{
    "cef_name": "dmac",
    "cef": 1,
    "cef_type": "default",
    "id": 1,
    "apply_filter": false
}
```

POST

Get a particular `indicator_cef_filter` record by ID.

Request parameters

Field	Required	Type	Description
apply_filter		boolean	Returns `true` if the associated CEF record will be filtered out during indicator creation.

JSON request

```
<div>
{
    "apply_filter": true
}
```

/rest/license

`/rest/license`
Automate loading your Splunk SOAR license.

POST

Automate loading your Splunk SOAR license.

JSON request

```
   "license":"<license>"
```

```
}
```

License formatting

The license must be a single line with the `\n` character encoded for new lines, as in the following example:

```
"license":"-----------------------BEGIN LICENSE-----------------------
-\nUVpONWpVREV1RXl5WWlvRlMrZDF4T2JYcW1mRkttSGRKZmRPZUNvYWo5bm5Q\nb3hsYWcwRkNNYTJOYUwzdm5WaVhodGZNenFzOVZaSUlWd
WtJdFl2THlQU2xm\nVGlYRlRCRy95V2NlUDh1d25XUFJNK2lhNWtmNWNnNlVRR3YzU01FYU8rSWt1\nN3plcDBBSlZwNlpZcTMzMHlwSzA2OWZDUFZm
... "
```

Aggregation rules endpoints

REST Aggregation Rules

Create or update an aggregation rule.

/rest/aggregation_rule

Create an aggregation rule.

Syntax

```
https://<username>:<password>@<host>/rest/aggregation_rule
```
POST

Create an aggregation rule.

Request string
An argument string must include the following fields.

Field	Required	Type	Description
rule	required	JSON object	This contains the main body of the aggregation rule, with 'name', and 'group' as top level keys. In 'group' each key is a CEF field which will be checked for matches, within this there is method and the may-be-required regex. If method is set to "Exact" then no regex value is required, and the aggregation will be done on exact match. If method is set to "Regex" the 'regex' value will be required to specify the regex pattern to be matched on, please see https://pypi.org/project/regex/ for more information on allowed regex patterns.
label	required	string	Only artifacts coming in with this label will be checked for aggregation rule matches. This field must be different than the destination label.
destination_label	required	string	Once a match has been found the artifact will be added to a container with this label. This field must be different than label.
tenants	optional	JSON array of integers	If applicable, the list of tenant ids which should use this rule to aggregate incoming data.

Example request
You can add an exact method by supplying a JSON formatted body.

```
curl -k -u admin:changeme https://localhost/rest/aggregation_rule \
-d '{
        "rule": {
                "name": "test rule exact",
                "group": {
                        "destinationAddress": {
                                "method": "Exact"
                        }
                }
        },
        "label": "phishing",
        "destination_label": "spear_phishing"
}'
```

Example request

You can add a regex method by supplying a JSON formatted body.

```
curl -k -u admin:changeme https://localhost/rest/aggregation_rule \
-d '{
        "rule": {
                "name": "test rule regex",
                "group": {
                        "sourceAddress": {
                                "method": "Exact"
                        },
                        "destinationAddress": {
                                "method": "Regex",
                                "regex": "10\.10\.6.*"
                        }
                }
        },
        "label": "phishing",
        "destination_label": "spear_phishing",
        "tenants": [12, 43]
}'
```

/rest/aggregation_rule/<aggregation_rule_id>

Update an existing aggregation rule.

Syntax

```
https://<username>:<password>@<host>/rest/aggregation_rule/<aggregation_rule_id>
```

Usage details

Optionally, you can leave off the aggregation_rule_id, but then it must be included in the request body. This facilitates bulk updates, passing a list of JSON objects each containing the appropriate Id. Special fields used for update are included below.

POST

Update an existing aggregation rule.

Request string

An argument string must include the following fields.

Field	Required	Type	Description
add_tenants	optional	JSON array of integers	Used in the same way as the 'tenants' field, however this will not remove any tenants which do not appear in the list.
remove_tenants	optional	JSON array of integers	The opposite of the 'add_tenants' field listed above, instead of replacing tenants with the tenant list, this will only remove those in the list.

Example request

Update tenant Ids 6 and 34.

```
curl -k -u admin:changeme https://localhost/rest/aggregation_rule \
-d '{
  "id": 100,
  "add_tenants": [6, 34]
}'
```

Example request

Remove tenant Ids 6 and 201

```
curl -k -u admin:changeme https://localhost/rest/aggregation_rule \
-d '{
  "id": 100,
  "remove_tenants": [6, 201]
}'
```

Example response

A successful POST will return back a success indicator and the ID of the newly created rule.

```
{
    "id": 100,
    "success": true
}
```

App endpoints

REST App

Install Splunk SOAR (On-premises) apps and see which Python version the apps are.

/rest/app

Syntax

```
https://<username>:<password>@<host>/rest/app
```

Usage details

The account used must have permissions to edit Apps.

POST

Install a Splunk SOAR (On-premises) app.

Example request

You can submit an HTTP POST to the following endpoint with a base64 encoded tarball or installer package.

```
curl -k -u admin:pwd https://localhost/rest/app \
-d '{
    "app": "<BASE64 ENCODED TARBALL OR INSTALLER PACKAGE>"
}
```

Success example response

A successful response includes the import success message and the success true status.

```
{
    "message": "App successfully imported.",
    "success": true
}
```

Failure example response

A failed response includes the import failure message and the failed true status.

```
{
    "failed": true,
    "message": "App install failed."
}
```

Example Python request

Python example of how the API can be called.

```
import json
import base64
import requests
file_contents = open('/path/to/myapp.tgz', 'rb').read()
encoded_contents = base64.b64encode(file_contents)
```

```
payload = {'app': encoded_contents}
requests.post('https://phantom.mycompany.com/rest/app',
                auth=('admin', PASSWORD),
                data=json.dumps(payload))
```

The app file can be a supported installer package such as a tarball or .rpm file. The app can be updated by simply POSTing the new app in the exact same manner as the original install.

DELETE

Delete a Splunk SOAR (On-premises) app.

Example request

Delete the app with Id 151.

```
curl -k -u admin:changeme https://localhost/rest/app/151 -X DELETE
```

/rest/app/<app-id>

Get the details of the specified app, including the Python version number.

Syntax

```
https://<username>:<password>@<host>/rest/app/<app-id>
```

Usage details

The account used must have permissions to view Apps.

GET

See the Python version number of the specified App.

Example request

Get the details of app id 112.

```
curl -k -u admin:changeme https://localhost/rest/app/112?pretty -G -X GET
```

Example response

A successful GET will return back a JSON formatted list of key names and their values.

```
{
  "app_config_render": null,
  "product_version_regex": ".*",
  "python_version": "3.6",
  "uber_view": null,
  "disabled": false,
  "logo": "logo_abuseipdb.svg",
  "install_time": "2019-07-15T01:31:42.560831Z",
  "id": 112,
  "logo_dark": "logo_abuseipdb_dark.svg",
  "rest_handler": null,
  "appname": "-",
  "_pretty_invalid_assets": [],
  "version": 1,
  "_pretty_actions": [
    {
      "description": "Report an IP for abusive behavior",
      "name": "post ip"
    },
    {
      "description": "Queries IP info",
      "name": "lookup ip"
    },
    {
      "description": "Validate the asset configuration for connectivity using supplied configuration",
      "name": "test connectivity"
    }
  ],
  "app_version": "1.0.9",
  "type": "reputation",
  "product_name": "AbuseIPDB",
  "description": "This app integrates with AbuseIPDB to perform investigative actions",
  "tags": [],
  "_pretty_asset_count": 1,
  "app_config": {},
  "_pretty_install_time": "Jul 15 at 01:31 AM",
  "configuration": {
    "api_key": {
      "required": true,
      "description": "API Key",
      "data_type": "password"
    }
  },
  "product_vendor": "AbuseIPDB",
  "publisher": "Splunk",
  "name": "AbuseIPDB",
  "release_tag": null,
  "consolidate_widgets": true,
  "appid": "52876771-17a7-45ad-8cc5-513bbd2172c5",
  "directory": "abuseipdb_52876771-17a7-45ad-8cc5-513bbd2172c5",
  "_pretty_dark_logo": "logo_abuseipdb_dark.svg",
  "require_auth_token": false,
  "main_module": "abuseipdb_connector.pyc",
  "known_versions": [
    "1.0.9"
  ]
}
```

The return values of note follow:

Field	Type	Description
app_version	string	The version of the app.

configuration	JSON object	Key value pairs for configuration. Required and optional values are defined by the Apps. See individual App documentation for more info. For example:
		`{...`
		`        "configuration": {`
		`                "api_key": {`
		`                        "required": true,`
		`                        "order": 0,`
		`                        "data_type": "password",`
		`                        "description": "API Key"`
		`                }`
		`        },`
		`...`

Field	Type	Description
		}
description	string	A brief description of the app.
id	string	The Id of the app.
install_time	string	The time that the app was installed, in epoch UTC format.
logo	string	The product logo in .svg or .png format.
logo_dark	string	The dark mode product logo in .svg or .png format.
name	string	Short name for the asset. Used when invoking an action on this asset.
product_name	string	Official name of the product. Used when invoking an action on this asset.
product_vendor	string	The name of the app vendor. Used when invoking an action on this asset.
publisher	string	The publisher of the app, such as Splunk SOAR (On-premises) or Splunk or SentinelOne.
python_version	string	Python version number.
tags	array of strings	0 or more tags associated with the asset. A simple string can also be used for a single tag. Optional, for use with Playbooks.
type	string	A brief categorization of this asset.

Approval endpoints

REST Approval

Single endpoint that provides details about existing approvals in the system.

/rest/approval

List of all approvals.

Syntax

```
https://<username>:<password>@<host>/rest/approval
```
GET

List of approvals.

Example request
Get a list of approvals.

```
curl -k -u admin:changeme https://localhost/rest/approval -G -X GET
```
Example response
A successful GET will return a 200 response, and a JSON formatted list of approvals.

```
{
"count": 5,
"data": [
{
"status": "expired",
"owner_type": "User",
"action_run": 23,
"playbook_run": 60,
"escalated_approval": null,
"name": "prompt_1",
"parent": null,
"node_guid": "eef4c48b-eef2-450e-a1b7-e90d2ef26fed",
"start_time": "2019-07-16T23:22:39.149000Z",
"close_time": "2019-07-16T23:52:39.247000Z",
"id": 1,
"due_time": "2019-07-16T23:52:39.115000Z",
"version": 1,
"jitc": {},
"asset": null,
"owner": 13,
"message": "pending-manual-action",
"type": "manual",
"display": true,
"responses": []
},
{
"status": "expired",
"owner_type": "User",
"action_run": 50,
"playbook_run": 66,
"escalated_approval": null,
"name": "task_1",
"parent": null,
"node_guid": "eef4c48b-eef2-450e-a1b7-e90d2ef26fed",
"start_time": "2019-07-29T23:28:43.149000Z",
"close_time": "2019-07-29T23:58:43.209000Z",
"id": 2,
"due_time": "2019-07-29T23:58:43.118000Z",
"version": 1,
"jitc": {},
"asset": null,
"owner": 1,
"message": "pending-manual-action",
"type": "manual",
"display": true,
"responses": []
},
{
"status": "pending",
"owner_type": "Action Reviewer",
"action_run": 51,
"playbook_run": 67,
"escalated_approval": null,
"name": "approval for add tag",
"parent": null,
"node_guid": "eef4c48b-eef2-450e-a1b7-e90d2ef26fed",
"start_time": "2019-08-19T05:38:05.728000Z",
"close_time": null,
"id": 3,
"due_time": "2019-08-20T05:38:05.726000Z",
"version": 1,
"jitc": {},
"asset": null,
"owner": 1,
"message": "pending-approval",
"type": "parameter",
"display": true,
"responses": []
},
{
"status": "pending",
"owner_type": "Action Reviewer",
"action_run": 52,
"playbook_run": 68,
"escalated_approval": null,
"name": "approval for add tag",
"parent": null,
"node_guid": "eef4c48b-eef2-450e-a1b7-e90d2ef26fed",
"start_time": "2019-08-19T05:38:58.723000Z",
"close_time": null,
"id": 4,
"due_time": "2019-08-20T05:38:58.721000Z",
"version": 1,
"jitc": {},
"asset": null,
"owner": 1,
"message": "pending-approval",
"type": "parameter",
"display": true,
"responses": []
```

```
},
{
"status": "pending",
"owner_type": "Action Reviewer",
"action_run": 53,
"playbook_run": 69,
"escalated_approval": null,
"name": "approval for activate device",
"parent": null,
"node_guid": "eef4c48b-eef2-450e-a1b7-e90d2ef26fed",
"start_time": "2019-08-19T16:25:03.062000Z",
"close_time": null,
"id": 5,
"due_time": "2019-08-20T16:25:03.060000Z",
"version": 1,
"jitc": {},
"asset": null,
"owner": 1,
"message": "pending-approval",
"type": "parameter",
"display": true,
"responses": []
}
],
"num_pages": 1
}
```

/rest/approval/<id>

Get the data of one approval.

Syntax

```
https://<username>:<password>@<host>/rest/approval/<id>
```

GET

List the approval data from one container Id.

Example request
Get a list of approvals.

```
curl -k -u admin:changeme https://localhost/rest/approval/1 -G -X GET
```

Example response
A successful GET will return a 200 response, and a JSON formatted list of data for one container Id.

```
{
        "status": "expired",
        "owner_type": "User",
        "action_run": 9,
        "playbook_run": 59,
        "escalated_approval": null,
        "name": "task_1",
        "parent": null,
        "node_guid": "9a8092d6-c3ad-4c61-b92a-005bb179cfc6",
        "start_time": "2020-01-22T19:39:43.239000Z",
        "close_time": "2020-01-22T20:09:43.295000Z",
        "id": 1,
        "due_time": "2020-01-22T20:09:43.221000Z",
        "version": 1,
        "jitc": {},
        "asset": null,
        "owner": 1,
        "message": "pending-manual-action",
        "type": "manual",
        "display": true,
        "responses": []
}
```

/rest/approval/<id>/detail_summary_view

List details of approvals for a particular container.

Syntax

```
https://<username>:<password>@<host>/rest/approval/<id>/detail_summary_view
```

GET

List details of approvals for a container where 21 is the approval ID in the example request.

Example request

List details of approvals.

```
curl -k -u admin:changeme https://localhost/rest/approval/21/detail_summary_view -G -X GET
```

Example response

A successful GET for approvals notification type will return a 200 response, and a JSON formatted list of details.

```
{
    "update_time": "2019-08-19T21:43:58.892936Z",
    "container_id": 291,
    "time_left": 80128.535132,
    "next_owner": null,
    "action_name": "user initiated post ip action",
    "due_time": "2019-08-20T20:05:57.814000Z",
    "asset": {
        "action_whitelist": {},
        "validation": {},
        "tenants": [],
        "description": "Default Asset Configuration for AbuseIPDB",
        "tags": [],
        "type": "reputation",
        "primary_voting": 0,
        "product_version": "",
        "effective_user": 2,
        "product_name": "AbuseIPDB",
        "disabled": false,
        "token": null,
        "version": 1,
        "secondary_voting": 0,
        "configuration": {
            "api_key": "r56jEhzRlV
/TR9CWLzDgN0GtxWrYQskkOl5ypVGUCNu1KKfy5f9EA40TY2piQLKCL040OtANINfTtV3vWF5kmElSRfHpb275bkN7didzCPpgpLg0PincyjONjA7P+
        },
        "product_vendor": "AbuseIPDB",
        "id": 70,
        "name": "abuse_ip_db"
    },
    "action_type": "post ip",
    "container_name": "Testcases.000100-Rest.000230-Custom Status.000200-Custom Status Severity Generator",
    "owner": "admin",
    "notification_type": "approvals",
    "type": "asset",
    "notification_targets": [{
        "app_id": 152,
        "parameters": [{
            "comment": "ddd",
            "ip": "3.3.3.33",
            "categories": "dd"
        }],
        "assets": [
            70
        ]
    }]
}
```

Example response

A successful GET for prompts notification type will return a 200 response, and a JSON formatted list of details.

```
{
    "playbook_repo": "local",
    "update_time": "2019-08-19T21:58:03.846035Z",
    "playbook_name": "pb-prompt",
    "container_id": 292,
    "time_left": 1758.571971,
    "next_owner": null,
    "action_name": "prompt_1",
    "due_time": "2019-08-19T22:28:03.817000Z",
    "asset": null,
    "action_type": "prompt",
    "container_name": "Testcases.000100-Rest.000230-Custom Status.000200-Custom Status Severity Generator",
    "owner": "admin",
    "notification_type": "prompts",
    "type": "manual",
    "notification_targets": [{
        "app_id": 0,
        "parameters": [{
            "to": "root@localhost",
            "message": "test",
            "mins_to_act": 30,
            "user_ids": [
                1
            ],
            "response_types": [{
                "prompt": "",
                "options": {
                    "type": "message"
                }
            }]
        }],
        "assets": []
    }]
}
```

Example response

A successful GET for manual tasks notification type will return a 200 response, and a JSON formatted list of details.

```
{
    "update_time": "2019-08-19T22:04:59.289861Z",
    "container_id": 293,
    "time_left": 3383.812224,
    "next_owner": null,
    "action_name": "user initiated task-18172",
    "due_time": "2019-08-19T23:04:59.240000Z",
    "asset": null,
    "action_type": "task",
    "container_name": "Testcases.000100-Rest.000230-Custom Status.000200-Custom Status Severity Generator",
    "owner": "admin",
    "notification_type": "manual tasks",
    "type": "manual",
    "notification_targets": []
}
```

Example response

A successful GET for action reviewers notification type will return a 200 response, and a JSON formatted list of details.

```
{
    "playbook_repo": "local",
    "update_time": "2019-08-19T22:14:06.436276Z",
    "playbook_name": "pb-reviewer",
    "container_id": 294,
    "time_left": 78412.135004,
    "next_owner": null,
    "action_name": "geolocate_ip_1",
    "due_time": "2019-08-20T20:05:58.356000Z",
    "asset": null,
    "action_type": "geolocate ip",
    "container_name": "Testcases.000100-Rest.000230-Custom Status.000200-Custom Status Severity Generator",
    "owner": "admin",
    "notification_type": "action reviews",
    "type": "parameter",
    "notification_targets": [{
        "app_id": 124,
        "parameters": [{
            "ip": "2.3.2.22"
        }],
        "assets": [
            2
        ]
    }]
}
```

The return values of note follow:

Field	Type	Description
asset	JSON Object	Can be empty depending on the notification type and if it contains an asset. See REST Assets for further information about assets.
container_id	String	The container Id of the playbook action run.
due_time	String	Time (UTC) when this action is due (time at which the SLA expires/expired).
next_owner	String	The next owner for an approval, such as admin.
notification_targets	JSON Object	JSON object containing a variety of parameters entered in response to prompt.
notification_type	String	prompts, approvals, manual tasks, action reviews.

Field	Type	Description
owner	String	The current owner's display name, such as admin.
playbook_name	String	The playbook name.
playbook_repo	String	The name of the the playbook repository.
prompt	String	The options available to respond to a prompt such as: • list: a list of pre-selected strings • message: a user-inputted string • range: a number in the range specified by the prompt response It returns a dictionary that organizes the response answer percentage by response.
time_left	String	The due time minus the current time, in seconds.
type	String	Mapping for prettifying notification types, such as: • prompt -> prompts • asset -> approvals • task -> manual tasks • parameter -> action reviews

The response varies depending on the notification type, which is a sub-type of approvals. The output is similar to `/rest/notification/<id>/detail_summary_view` used for mobile. See REST Notification.

Artifact endpoints

REST Artifact

Artifacts are objects that are associated with a Container and serve as corroboration or evidence related to the Container. See Add artifacts from a container to a case in *Use Splunk SOAR (On-premises)*.

/rest/artifact

To optimize performance when creating multiple artifacts, first create the container, then create all artifacts except the last with run_automation set to False, and then create the last artifact with run_automation set to True. This will cause automation such as active playbooks to run only after all artifacts have been added. To be even more efficient, you can create the container and artifacts in a single POST (see REST container documentation) in which case you should not set run_automation at all, as Splunk SOAR (On-premises) will automatically set run_automation after the last artifact is created.

Syntax

```
https://<username>:<password>@<host>/rest/artifact
```
POST

Create a new artifact.

Request string
An argument string must include the following fields: `container_id`.

Field	Required	Type	Description
cef	optional	Javascript object.	Contains standard fields available in the Common Event Format.
cef_types	optional	Javascript object.	Allows association of "contains" information to custom CEF fields. Object keys should be keys in the "cef" object. Values should be a list of strings where the strings are standard "contains" data types such as "ip" or "pid" etc.
container_id	required	integer	The artifact will "belong" to this container.
data	optional	JSON Object	Custom data field.
description	optional	string	A textual description of the artifact.
end_time	optional	ISO 8601 formatted timestamp	Date and time (in UTC) when the behavior tracked by the container stopped.
ingest_app_id	optional	integer or String	ID of the app which produced the artifact. Name of the app can also be provided.
kill_chain	optional	string	Cyber kill chain. One of • Reconnaissance • Weaponization • Delivery • Exploitation • Installation

Field	Required	Type	Description
			• Command & Control • Actions on Objectives
label	optional	string	The label classifies the artifact. Typically the label will be one of: • event • net flow
name	optional	string	A human friendly name for the artifact.
owner_id	optional	integer or string	ID of the user who should own the artifact. Username can also be used.
run_automation	optional	boolean	Not an artifact data field: This parameter instructs Splunk SOAR (On-premises) to run automation upon creation or update of the artifact, and defaults to True.
severity	optional	string	The severity level of the artifact you are adding. Helps to determine the SLA applied to Actions related to the container. Either one of Low, Medium, or High or else a custom severity name set by an administrator. If the severity level of the artifact is higher than the current severity level of the container, then the container's severity will be changed to match the artifact. E.g. if you add a high severity artifact to a medium severity container, the container will be changed to severity high. You can set a container's severity to Low, Medium, or High with this endpoint even if those severity names have been deleted by the administrator.
source_data_identifier	optional	string	ID which can be used to find this container in the source product. (e.g. the container was retrieved from a SIEM, this is the ID in the SIEM)
start_time	optional	ISO 8601 formatted timestamp	Date and time (in UTC) when the behavior tracked by the container started.
tags	optional	Array of strings	0 or more tags associated with the asset. A simple string can also be used for a single tag.
type	optional	string	Helps to identify the content of this artifact. Typically a string such as "network" or "host" etc.

Response

A success or failure message.

Example request

Add a new artifact as follows:

```
curl -k -u admin:changeme https://localhost/rest/artifact \
-d '{
        "asset_id": 10,
        "cef": {
                "ApplicationProtocol": "",
                "act": "",
                "app": "",
                "baseEventCount": "120",
                "bytesIn": "",
                "bytesOut": "",
                "cat": "",
                "cn1": "",
                "cn1Label": "",
                "cn2": "",
"cn2Label": "",
"cn3": "",
"cn3Label": "",
"cnt": "",
"cs1": "",
"cs1Label": "",
"cs2": "",
"cs2Label": "",
"cs3": "",
"cs3Label": "",
"cs4": "",
"cs4Label": "",
"cs5": "",
"cs5Label": "",
"cs6": "",
"cs6Label": "",
"destinationAddress": "",
"destinationDnsDomain": "",
"destinationHostName": "",
"destinationMacAddress": "",
"destinationNtDomain": "",
"destinationPort": "80",
"destinationProcessName": "",
"destinationServiceName": "",
"destinationTranslatedAddress": "",
"destinationTranslatedPort": "",
"destinationUserId": "",
"destinationUserName": "",
"destinationUserPrivileges": "",
"deviceAction": "",
"deviceAddress": "",
"deviceCustomDate1": "",
"deviceCustomDate1Label": "",
"deviceCustomDate2": "",
"deviceCustomDate2Label": "",
"deviceCustomNumber1": "",
"deviceCustomNumber1Label": "",
"deviceCustomNumber2": "",
"deviceCustomNumber2Label": "",
"deviceCustomNumber3": "",
"deviceCustomNumber3Label": "",
"deviceCustomString1": "",
"deviceCustomString1Label": "",
"deviceCustomString2": "",
"deviceCustomString2Label": "",
"deviceCustomString3": "",
"deviceCustomString3Label": "",
"deviceCustomString4": "",
"deviceCustomString4Label": "",
"deviceCustomString5": "",
"deviceCustomString5Label": "",
"deviceCustomString6": "",
"deviceCustomString6Label": "",
"deviceDirection": "",
"deviceDnsDomain": "",
"deviceEventCategory": "",
"deviceExternalId": "",
"deviceFacility": "",
"deviceHostname": "",
"deviceInboundInterface": "",
"deviceMacAddress": "",
"deviceOutboundInterface": "",
"deviceProcessName": "",
"deviceTranslatedAddress": "",
"dhost": "",
"dmac": "",
"dntdom": "",
"dpriv": "",
"dproc": "",
"dpt": "",
"dst": "103.230.84.239",
"duid": "",
"duser": "",
"dvc": "",
"dvchost": "",
"end": "",
"endTime": "",
"externalId": "",
"fileCreateTime": "2014-10-19 12:41:32",
"fileHash": "51020390505ecc8cf70456756399374219965 29f6d49decc53753e1335aeb574",
"fileId": "",
"fileModificationTime": "",
"fileName": "",
"filePath": "",
"filePermission": "",
"fileSize": "",
"fileType": "",
"fname": "",
"fsize": "",
"in": "",
"message": "",
"method": "",
"msg": "",
"oldfileCreateTime": "",
"oldfileHash": "",
"oldfileId": "",
"oldfileModificationTime": "",
"oldfileName": "",
"oldfilePath": "",
"oldfilePermission": "",
"oldfileType": "",
"oldfsize": "",
"out": "",
"proto": "",
"receiptTime": "",
"request": "",
"requestClientApplication": "",
"requestCookies": "",
"requestMethod": "",
"requestURL": "",
"rt": "",
"shost": "",
"smac": "",
"sntdom": "",
"sourceAddress": "10.10.0.201",
"sourceDnsDomain": "",
"sourceHostName": "",
"sourceMacAddress": "",
"sourceNtDomain": "",
"sourcePort": "4286",
"sourceServiceName": "",
"sourceTranslatedAddress": "",
"sourceTranslatedPort": "",
"sourceUserId": "",
                "sourceUserName": "",
                "sourceUserPrivileges": "",
                "spriv": "",
                "spt": "",
                "src": "",
                "start": "",
                "startTime": "09/09/2014 16:30:00",
                "suid": "",
                "suser": "",
                "transportProtocol": "",
                "my_custom_cef_field": "1.1.1.1"
        },
        "cef_types": {
                "my_custom_cef_field": ["ip"]
        },
        "container_id": 41,
        "data": {},
        "end_time": "2014-10-19T14:45:51.100Z",
        "label": "event",
        "run_automation": true,
        "severity": "high",
        "source_data_identifier": "4",
        "start_time": "2014-10-19T14:41:33.384Z",
        "tags": ["tag1", "tag2"],
        "type": "network"
}'
```

Example response

A successful POST will return back a success indicator and the ID of the newly created artifact.

```
{
    "id": 41,
    "success": true
}
```

Posting a JSON that is identical to an existing artifact will result in a duplication error. The response will also return the ID of the matching artifact.

```
{
    "existing_artifact_id": 41,
    "failed": true,
    "message": "artifact already exists"
}
```

Additional example

Example request

Create an artifact.

```
{
    "container_id": 2,
    "severity": "low",
    "label": "events",
    "cef": {
        "sourceAddress": "127.0.0.1"
    },
    "cef_types": {
        "sourceAddress": [
            "ip"
        ]
    },
    "name": "Ping event"
}
```

Asset endpoints

REST Asset

Assets are specific instances of physical or virtual devices within your organization such as servers, endpoints, routers, and firewalls. See About Splunk SOAR (On-premises).

/rest/asset

Syntax

```
https://<username>:<password>@<host>/rest/asset
```

POST

Create a new asset.

Request string
An argument string must include one of the following sets of fields:

- `app_id`
- `app_guid`
- `name`, `product_name`, `product_vendor`

For polling an asset, an argument string must include the following sets of fields:

- Either:
    - `app_id`
    - `app_guid`
    - `name`, `product_name`, `product_vendor`
- And:
    - `container_label`
    - `interval_mins`
    - `poll`
    - `start_time_epoch_utc`

Field	Required	Type	Description
app_id	required	string	The unique id for the App used by the asset. Used when invoking an action on this asset. Required unless you specify the `app_guid` or the `name`, `product_name`, `product_vendor`.
app_guid	required	string	The GUID for the App used by the asset. Used when invoking an action on this asset. Required unless you specify the `app_id` or the `name`, `product_name`, `product_vendor`.
configuration	optional	JSON Object	Key value pairs for asset configuration. Required and optional values are defined by Apps which support this asset. See individual App documentation for more information.
container_label	required for polling	string	Label for containers created by this asset. This will determine where the container will be found from the main menu. For example, using "incident" here will allow you to find the container by going to Home -> Incidents.
description	optional	string	Description of the asset.

Field	Required	Type	Description
interval_mins	required for polling	number	Polling frequency for new containers/artifacts in minutes. Not specified or null for assets that do not poll.
name	required	string	Short name for the asset. Used when invoking an action on this asset.
poll	required for polling	boolean	Set to "true" to enable polling.
primary_owners	optional	Array of integers	Contains the list of user IDs which are the primary owners of this asset.
primary_voting	optional	integer	Number of "approve" votes required for action to be executed. Use 0 to require all votes.
product_name	required	string	Name of the product which this asset describes. Must match the product_name provided by one or more Apps. Example: If this asset refers to a Windows Server 2008 machine and an App supports a product_name of "Server 2008" then that is the correct value.
product_vendor	required	string	Name of the vendor of the product which this asset describes. Must match the product_vendor provided by one or more Apps. Example: If this asset refers to a Windows Server 2008 machine and an App supports a product_vendor of "Windows" then that is the correct value.
secondary_users	optional	Array of integers	Contains the list of user IDs which are the secondary (backup) owners of this asset.
secondary_voting	optional	integer	Number of "approve" votes required for action to be executed for secondary users. Use 0 to require all votes.
start_time_epoch_utc	required for polling	number	First poll time as seconds since epoch UTC. Use null to poll immediately.
tags	optional	Array of strings	0 or more tags associated with the asset. A simple string can also be used for a single tag.
type	optional	string	A brief categorization of this asset.

action_whitelist	optional	dictionary	A dictionary object which contains actions and who is whitelisted to run the action. A star may be used in place of action name, users, or roles to indicate all. `;{"action": {"roles": ["7"]}}`
			`: indicates users in role 7 are approved to`
			`: run 'action'.`
			`; {"*": {"roles": ["7"], "users": ["6"]}`
			`: indicates users in role 7 and user 6 are`
			`: approved to run all actions.`
			`; {"action": {"users": ["*"]}`
			`: indicates all users are approved to`
			`: run 'action'.`

tenants	optional	Array of integers	Contains the list of tenant IDs which are to be associated with this asset. Only one tenant is allowed on ingestion assets, trying to add more will return an error. Tenants will only be added or removed based on the acting user's permissions, i.e. passing an empty list will remove all tenants from an asset for which the user has permission to edit.
_reserved_jitc	optional	JSON Object	This key should reside under "configuration". Keys in this object are the names of configuration parameters. These configuration parameters will not be stored in the database and must be provided each time actions are run by an asset owner or the action will fail. The values are always a literal "true". This can be used for sensitive asset credentials. At least one asset owner must be selected for actions to succeed if this feature is used. Cannot be used on assets that perform ingestion.

Response

A success or failure message.

Example request
Add a new asset as follows:

```
curl -k -u admin:changeme https://localhost/rest/asset \
-d '{
        "configuration": {
                "foo": "bar"
        },
        "description": "My Generic/Product asset.",
        "name": "my asset",
        "primary_owners": [12],
        "primary_voting": 0,
        "product_name": "Product",
        "product_vendor": "Generic",
        "secondary_users": [],
        "secondary_voting": 0,
        "tags": ["tag1", "tag2"],
        "type": "information service",
        "action_whitelist": {
                "generic action": {
                        "roles": ["7"],
                        "users": ["6"]
                }
        },
        "tenants": [23, 10, 9]
}'
```

Example response
A successful POST will return back a success indicator and the ID of the newly created asset.

```
{
    "id": 34,
    "success": true
}
```

Example request
Create an asset preconfigured and enabled for polling to ingest data:

```
curl -k -u admin:changeme https://localhost/rest/asset \
-d '{
  "configuration": {
    "foo": "bar",
    "ingest": {
        "container_label": incident",
        "interval_mins": 120,
        "poll":  true,
        "start_time_epoch_utc": null
    },
    ...
  },
  "description": "My Generic/Product asset.",
  ...
}'
```

Example response
A successful POST will return back a success indicator and the ID of the newly created asset.

```
{
    "id": 34,
    "success": true
}
```

If you use this API to update an asset, any value you do not submit in your POST is reset to its default value. Partial updates are not recommended.

Audit endpoints

REST Audit

You can access audit information for individual Users, Roles, Playbooks, and Containers. Or you can access all available audit information at once, with or without additional filtering.

/rest/audit

List all audit events.

Syntax

```
https://<username>:<password>@<host>/rest/audit
```

Usage details
By default, with no additional variables provided, this rest endpoint will list all audit events for the last 30 days. Like the rest endpoints for individual objects, you can specify "format", "sort", "start" and "end", which will still give you all categories of audit events. Changing the start or end times will change the default start of 30 days ago, or the default end time of now.

Because the rest endpoints for other individual objects inherently specified the object in question, there is no need to specify the set of objects you want audit events for. For the "all" endpoint, the default set is all objects. But you may wish to specify a subset to get just the events you want. As long as you do not specify any object set, you will get all. But as soon as you specify any object, you limit the results to just that object or set of objects.

GET

Request all audit information.

Request string
An argument string must include the following fields: `id`, where id is the number of the container. See Optional parameters for more information.

Example request
For objects specifying a single user, role, playbook, or container, these would be the equivalent of using the individual rest endpoints for the same id. All of theses pairs of URLs would produce the same results:

```
curl -k -u admin:changeme https://localhost/rest/audit?user=1 -G -X GET
curl -k -u admin:changeme https://localhost/rest/user/1/audit -G -X GET

curl -k -u admin:changeme https://localhost/rest/audit?role=6 -G -X GET
curl -k -u admin:changeme https://localhost/rest/role/6/audit -G -X GET

curl -k -u admin:changeme https://localhost/rest/audit?playbook=1 -G -X GET
curl -k -u admin:changeme https://localhost/rest/playbook/1/audit -G -X GET

curl -k -u admin:changeme https://localhost/rest/audit?container=1 -G -X GET
curl -k -u admin:changeme https://localhost/rest/container/1/audit -G -X GET
```

The difference is that with /rest/audit you can specify a wildcard (*) to get all objects of that type, or you can specify more than one type at a time, or you can specify a list of objects.

Example request
Specify that you want all playbook audit events at once by specifying `playbook=*` with this URL:

```
curl -k -u admin:changeme https://localhost/rest/audit?playbook=* -G -X GET
```

Example request
For "authentication" and "administration" events, you can only specify * since there are no individual objects to refer to, just those categories. These are effectively the "on" for including these events types. You can ask for multiple objects types at once by specifying multiple types in the URL:

```
curl -k -u admin:changeme https://localhost/rest/audit?authentication=*&administration=* -G -X GET
```

Example request
And finally, you can specify multiple individual IDs for the same object type by separating them with %1E, for example, to retrieve audit events for user IDs 1 and 2:

```
curl -k -u admin:changeme https://localhost/rest/audit?user=1%1E2 -G -X GET
```

/rest/audit optional filter

An additional, optional filter is provided to filter based on tenant mapping. If tenant is supplied, the results of all other requested audits will be filtered based on the provided tenant. Using this filter will still include records where no specific mapping exists, implying all tenants, so asset or administration changes that affect all tenants will be included, depending on the type of audit requested. Since this is a filter, * is not supported, as that would be the same as not including it. The filter takes either tenant id or tenant name, and supports multiple arguments using the %1E field separator. If the filter is not included, all records will be included, no matter which tenant they affect.

tenant: (tenant_id or tenant_name)

Example request
You can specify multiple individual IDs for the same object type by separating them with %1E, for example, to retrieve audit events for user IDs 1 and 2:

```
curl -k -u admin:changeme https://localhost/rest/audit?container&admnistration&tenant=4%1E2 -G -X GET
```

The example is retrieving all container and administration operations filtering for those which affect tenant 4 or tenant 2, which will include records affecting all tenants (since this would implicitly include tenant 2 or tenant 4).

/rest/ph_user/<id>/audit

Audit of /rest/ph_user/<userid>.

Syntax

```
https://<username>:<password>@<host>/rest/ph_user/<id>/audit
```

GET

Request the audit information related to a single user.

Request string
An argument string must include the following fields: `id`, where id is the number of the user. See Optional parameters for more information.

Example request
Request the audit information related to a single user, where 1 is the ID number of the user. For example, the "admin" user is usually ID 1. To see everything the admin user has done while auditing was enabled, do the following:

```
curl -k -u admin:changeme https://localhost/rest/ph_user/1/audit -G -X GET
```

Example response
A successful GET will return the audit of everything the admin user has done.

```
[
  {
    "AUDIT ID": 37,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": null,
    "NOTE": "GET:/rest/ph_user/1/audit",
    "OBJECT ID": 1,
    "OBJECT TYPE": "ph_user",
    "OLD VALUE": null,
    "RELATED OBJECT ID": null,
    "RELATED OBJECT TYPE": null,
    "TIME": "2017-07-06T00:47:20.692539Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 4,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "disabled",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "true",
    "NOTE": "local/list vms:Previous Record ID:56",
    "OBJECT ID": 57,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "false",
    "RELATED OBJECT ID": 56,
    "RELATED OBJECT TYPE": "playbook",
    "TIME": "2017-07-05T21:47:42.257788Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 3,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "comment",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "3",
    "NOTE": "local/list vms:Previous Record ID:56",
    "OBJECT ID": 57,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "2",
    "RELATED OBJECT ID": 56,
    "RELATED OBJECT TYPE": "playbook",
    "TIME": "2017-07-05T21:47:42.257748Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 2,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "change",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "fa3d71372ac87c1291208e25804af5f7749e79e4",
    "NOTE": "local/list vms:Previous Record ID:56",
    "OBJECT ID": 57,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "88d24966b067f7c7e02de1c208090f736b85dbdd",
    "RELATED OBJECT ID": 56,
    "RELATED OBJECT TYPE": "playbook",
    "TIME": "2017-07-05T21:47:42.257660Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 2,
    "AUDIT SOURCE": "Action Run",
    "CHANGED FIELD": "",
    "IP ADDRESS": "",
    "NEW VALUE": "",
    "NOTE": "local/list vms:whois_ip_1",
    "OBJECT ID": "",
    "OBJECT TYPE": "action",
    "OLD VALUE": "",
    "RELATED OBJECT ID": 1,
    "RELATED OBJECT TYPE": "container",
    "TIME": "2017-07-05T21:45:47.568000Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 1,
    "AUDIT SOURCE": "Action Run",
    "CHANGED FIELD": "",
    "IP ADDRESS": "",
    "NEW VALUE": "",
    "NOTE": "local/list vms:list_vms_1",
    "OBJECT ID": "",
    "OBJECT TYPE": "action",
    "OLD VALUE": "",
    "RELATED OBJECT ID": 1,
    "RELATED OBJECT TYPE": "container",
    "TIME": "2017-07-05T21:45:37.077000Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 1,
    "AUDIT SOURCE": "Playbook Run",
    "CHANGED FIELD": "",
    "IP ADDRESS": "",
    "NEW VALUE": "",
    "NOTE": "local/list vms:run",
    "OBJECT ID": 56,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "",
    "RELATED OBJECT ID": 1,
    "RELATED OBJECT TYPE": "container",
    "TIME": "2017-07-05T21:45:36.925000Z",
    "USER": "admin",
    "USER ID": 1
  }
]
```

A large number of auditing events have been snipped from this example for the sake of brevity. With all auditing enabled, the audit records can be extensive. It should be noted that accessing audit logs is itself an auditable action. So if you use a user like admin to access the admin audit record, then you have just added another audit item to the log. Therefore, each subsequent call will return a slightly larger set of records than the one before.

Also note that the AUDIT ID numbers (alone) are not necessarily unique. This is because the audit record comes from more than one database table, each of which has their own set of row numbers. It's the combination of AUDIT SOURCE and AUDIT ID together that make up the unique identifier. The example contains both "Playbook Run":"1" and "Action Run":"1", which are two different item's event though they both have an AUDIT ID of 1, because they have different AUDIT SOURCE values. You will not have two "Playbook Run":"1" audit events, or any audit entries with both the same ID and SOURCE.

/rest/role/<id>/audit

Audit of /rest/role/<role_id>.

Syntax

```
https://<username>:<password>@<host>/rest/role/<id>/audit
```

Usage

Requesting role audit events returns two types of events. First are changes to the role itself, such as creating the role, or changes to permissions in it. Second are events for users currently in the role at the time of the API call. If a user had that role yesterday, but has since been removed, when you request audit events for that role the removed user's events will not appear. For user events, the role here is acting like a group of users. If the role has no users assigned, you will only get the modifications to the role itself.

GET

Request the audit information related to a role.

Request string
An argument string must include the following fields: `id`, where id is the number of the role. See Optional parameters for more information.

Example request
Request the audit information related to a single user, where 7 is the ID number of the user. To request the format in "csv" instead of the default "json", do the following:

```
curl -k -u admin:changeme https://localhost/rest/role/7/audit?format=csv -G -X GET
```

Example response
A successful GET will return a user named "herman" who is the only user in role number 7. Also included is the herman login, and then log authorization errors because that user does not have the necessary privileges to access the objects in question. The example log has been truncated.

```
AUDIT SOURCE,AUDIT ID,TIME,USER ID,USER,IP ADDRESS,OBJECT ID,OBJECT TYPE,RELATED OBJECT TYPE,RELATED OBJECT
ID,NOTE,CHANGED FIELD,OLD VALUE,NEW VALUE
Audit,62,2017-07-06 03:06:04.086355+00:00,3,herman,10.10.0.240,,authorization,,,users_roles:view:Failed,,,
Audit,61,2017-07-06 03:06:04.083012+00:00,3,herman,10.10.0.240,,authorization,,,containers:view:Failed,,,
Audit,60,2017-07-06 03:06:03.800788+00:00,3,herman,10.10.0.240,,authentication,,,login:Success,,,
```

/rest/playbook/<id>/audit

Audit of /rest/playbook/<id>.

Syntax

```
https://<username>:<password>@<host>/rest/playbook/<id>/audit
```
GET

Request the audit information related to a playbook.

Request string
An argument string must include the following fields: `id`, where id is the number of the playbook. See Optional parameters for more information.

Example request
Request the audit information related to a playbook, where 59 is the ID number of the playbook. To request an ascending sort with oldest items first, do the following:

```
curl -k -u admin:changeme https://localhost/rest/playbook/59/audit?sort=asc -G -X GET
```
Example response
A successful GET will return back a user named "herman" who is the only user in role number 7. Also included is the user's login attempts, and then log authorization errors because that user does not have the needed privileges to access the objects in question. The example log has been truncated.

```
[
  {
    "AUDIT ID": 149,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "change",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "24591582fdaa19b98d4d2681e5098ce712167cc4",
    "NOTE": "local/whois ip:",
    "OBJECT ID": 59,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": null,
    "RELATED OBJECT ID": null,
    "RELATED OBJECT TYPE": null,
    "TIME": "2017-07-06T03:20:28.976883Z",
    "USER": "admin",
    "USER ID": 1
  },
{
  "AUDIT ID": 150,
  "AUDIT SOURCE": "Audit",
  "CHANGED FIELD": "disabled",
  "IP ADDRESS": "10.10.0.240",
  "NEW VALUE": "false",
  "NOTE": "local/whois ip:",
  "OBJECT ID": 59,
  "OBJECT TYPE": "playbook",
  "OLD VALUE": "true",
  "RELATED OBJECT ID": null,
  "RELATED OBJECT TYPE": null,
  "TIME": "2017-07-06T03:20:29.055953Z",
  "USER": "admin",
  "USER ID": 1
},
{
  "AUDIT ID": 151,
  "AUDIT SOURCE": "Audit",
  "CHANGED FIELD": "passed_validation",
  "IP ADDRESS": "10.10.0.240",
  "NEW VALUE": "true",
  "NOTE": "local/whois ip:",
  "OBJECT ID": 59,
  "OBJECT TYPE": "playbook",
  "OLD VALUE": "false",
  "RELATED OBJECT ID": null,
  "RELATED OBJECT TYPE": null,
  "TIME": "2017-07-06T03:20:29.056054Z",
  "USER": "admin",
  "USER ID": 1
},
{
  "AUDIT ID": 2,
  "AUDIT SOURCE": "Playbook Run",
  "CHANGED FIELD": "",
  "IP ADDRESS": "",
  "NEW VALUE": "",
  "NOTE": "local/whois ip:run",
  "OBJECT ID": 59,
  "OBJECT TYPE": "playbook",
  "OLD VALUE": "",
  "RELATED OBJECT ID": 1,
  "RELATED OBJECT TYPE": "container",
  "TIME": "2017-07-06T03:20:38.452000Z",
  "USER": "admin",
  "USER ID": 1
},
{
  "AUDIT ID": 3,
  "AUDIT SOURCE": "Action Run",
  "CHANGED FIELD": "",
  "IP ADDRESS": "",
  "NEW VALUE": "",
  "NOTE": "local/whois ip:whois_ip_1",
  "OBJECT ID": "",
  "OBJECT TYPE": "action",
  "OLD VALUE": "",
  "RELATED OBJECT ID": 1,
  "RELATED OBJECT TYPE": "container",
  "TIME": "2017-07-06T03:20:38.466000Z",
  "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 153,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "change",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "468ea286a5116c5f924401e802cf420df2f74c97",
    "NOTE": "local/whois ip:Previous Record ID:59",
    "OBJECT ID": 60,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "24591582fdaa19b98d4d2681e5098ce712167cc4",
    "RELATED OBJECT ID": 59,
    "RELATED OBJECT TYPE": "playbook",
    "TIME": "2017-07-06T03:22:23.436100Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 154,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "comment",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "Changed IP",
    "NOTE": "local/whois ip:Previous Record ID:59",
    "OBJECT ID": 60,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "Initial version",
    "RELATED OBJECT ID": 59,
    "RELATED OBJECT TYPE": "playbook",
    "TIME": "2017-07-06T03:22:23.436231Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 155,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "disabled",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "true",
    "NOTE": "local/whois ip:Previous Record ID:59",
    "OBJECT ID": 60,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "false",
    "RELATED OBJECT ID": 59,
    "RELATED OBJECT TYPE": "playbook",
    "TIME": "2017-07-06T03:22:23.436273Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 156,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "old_name",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "whois ip",
    "NOTE": "local/whois ip:Previous Record ID:59",
    "OBJECT ID": 60,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": null,
    "RELATED OBJECT ID": 59,
    "RELATED OBJECT TYPE": "playbook",
    "TIME": "2017-07-06T03:22:23.436310Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 157,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "passed_validation",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "false",
    "NOTE": "local/whois ip:Previous Record ID:59",
    "OBJECT ID": 60,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "true",
    "RELATED OBJECT ID": 59,
    "RELATED OBJECT TYPE": "playbook",
    "TIME": "2017-07-06T03:22:23.436343Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 158,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "version",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "2",
    "NOTE": "local/whois ip:Previous Record ID:59",
    "OBJECT ID": 60,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "1",
    "RELATED OBJECT ID": 59,
    "RELATED OBJECT TYPE": "playbook",
    "TIME": "2017-07-06T03:22:23.436377Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 159,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "disabled",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "true",
    "NOTE": "local/whois ip:",
    "OBJECT ID": 59,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "false",
    "RELATED OBJECT ID": null,
    "RELATED OBJECT TYPE": null,
    "TIME": "2017-07-06T03:22:23.508084Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 160,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "disabled",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "false",
    "NOTE": "local/whois ip:",
    "OBJECT ID": 60,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "true",
    "RELATED OBJECT ID": null,
    "RELATED OBJECT TYPE": null,
    "TIME": "2017-07-06T03:22:23.513248Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 161,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "passed_validation",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "true",
    "NOTE": "local/whois ip:",
    "OBJECT ID": 60,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": "false",
    "RELATED OBJECT ID": null,
    "RELATED OBJECT TYPE": null,
    "TIME": "2017-07-06T03:22:23.513313Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 162,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": null,
    "NOTE": "GET:/rest/playbook/59/audit",
    "OBJECT ID": 59,
    "OBJECT TYPE": "playbook",
    "OLD VALUE": null,
    "RELATED OBJECT ID": null,
    "RELATED OBJECT TYPE": null,
    "TIME": "2017-07-06T03:22:32.051042Z",
    "USER": "admin",
    "USER ID": 1
  }
]
```

A small number of operations were performed on this playbook to illustrate what the API returns, but it generated a fair number of events. A playbook called "whois ip" was created and saved. That's audit IDs 149 through 151. Then it was run, which shows as playbook run 2 and action run 3. And then it was changed and saved again, which is shown in audit ID 153 and higher. The playbook started out as ID 59, which is what we requested in the API call. When it was changed and saved again, it became ID 60. Those changes are reflected in the audit events. If you ask for either playbook 59 or 60 from then on, you will get the audit information for all versions of that playbook. You do not have to track the changes and ask for subsequent versions, any version of it will work.

/rest/container/<id>/audit

Audit of /rest/container/<container_id>.

Syntax

```
https://<username>:<password>@<host>/rest/container/<id>/audit
```

GET

Request the audit information related to a container.

Request string
An argument string must include the following fields: `id`, where id is the number of the container. See Optional parameters for more information.

Example request
Request the audit information related to a playbook, where 59 is the ID number of the playbook. To request events before a particular end date, and an ascending sort with oldest items first, do the following:

```
curl -k -u admin:changeme https://localhost/rest/container/1/audit?end=2017-07-07&sort=asc -G -X GET
```

Example response
A successful GET will return back the event dates of 2017-07-06 for the container. This is because the example request used only the date without specifying a time of day. Python substitutes a time of 00:00:00, or the very beginning of the day if you leave off the time. If the example was meant to specify the end of the day on 2017-07-06, it would have to specify 2017-07-06T23:59:59.999999Z, or just use 2017-07-07 as shown.

```
[
  {
    "AUDIT ID": 365,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": null,
    "NOTE": "GET:/rest/container/1/comments?page=0&page_size=20&sort=time&order=desc&pretty=true",
    "OBJECT ID": 1,
    "OBJECT TYPE": "container",
    "OLD VALUE": null,
    "RELATED OBJECT ID": null,
    "RELATED OBJECT TYPE": null,
    "TIME": "2017-07-06T22:16:04.602689Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 1,
    "AUDIT SOURCE": "Action Run",
    "CHANGED FIELD": "",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "",
    "NOTE": "user initiated list vms action",
    "OBJECT ID": "",
    "OBJECT TYPE": "action",
    "OLD VALUE": "",
    "RELATED OBJECT ID": 1,
    "RELATED OBJECT TYPE": "container",
    "TIME": "2017-07-06T22:17:34.392168Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 2,
    "AUDIT SOURCE": "Action Run",
    "CHANGED FIELD": "",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": "",
    "NOTE": "user initiated whois ip action",
    "OBJECT ID": "",
    "OBJECT TYPE": "action",
    "OLD VALUE": "",
    "RELATED OBJECT ID": 1,
    "RELATED OBJECT TYPE": "container",
    "TIME": "2017-07-06T22:18:03.566282Z",
    "USER": "admin",
    "USER ID": 1
  },
  {
    "AUDIT ID": 368,
    "AUDIT SOURCE": "Audit",
    "CHANGED FIELD": "",
    "IP ADDRESS": "10.10.0.240",
    "NEW VALUE": null,
    "NOTE": "GET:/rest/container/1/artifacts?page=0&page_size=5&sort=id&order=desc&pretty=true",
    "OBJECT ID": 1,
    "OBJECT TYPE": "container",
    "OLD VALUE": null,
    "RELATED OBJECT ID": null,
    "RELATED OBJECT TYPE": null,
    "TIME": "2017-07-06T22:18:14.021982Z",
    "USER": "admin",
    "USER ID": 1
  },
]
```
The results in the example response have been heavily truncated, because using the web UI to look at the container generates a lot of records like Audit 365 and Audit 368 as shown. You can also see two action runs. You can see that accessing the rest endpoint for a container is logged as an audit event if you have that auditing enabled. That will include the rest call you made to get these results. So every time you make this rest call, there will be at least one new event.

Optional parameters

The optional parameters for the audit REST API endpoints follow.

Audit Type	Optional Parameters
ph_user, role, playbook, container (`/rest<type>/<id>/audit`)	format: (json by default or csv) sort: (desc by default or asc) start: [datetime] 30 days prior to the current time. end: [datetime] Accepted date formats are ISO 8601 date or date and time.
ALL (`/rest/audit`)	format: (json or csv) sort: (desc or asc) start: [datetime] 30 days prior to the current time. end: [datetime] Accepted date formats are ISO 8601 date or date and time. user: (user_id or *) role: (role_id or *) authentication: [*] administration: [*] playbook: (playbook_id, repo/pb_name, or *) container: (container_id, container_name, or *)

Audit Type	Optional Parameters
	Additional Info: • For all types, multiple values may be selected by separating with %1E (ASCII field separator). • For playbook and container, a combination of name/id is possible, again using %1E as a separator.

CEF endpoints

REST CEF

Splunk SOAR (On-premises) uses the Common Event Format (CEF). CEF is a system of key:value pairs for key pieces of information about an artifact. The value is often referred to as the contains as shorthand.

/rest/cef

Get a list of available CEF.

Syntax

```
https://<username>:<password>@<host>/rest/cef
```

Usage details

Use parameters to get additional pages or sorting. All default CEFs have a type of "default." Custom CEFs have the type "custom." Only custom CEFs are mutable. CEFs with the type "default" cannot be modified.

GET

Get a list of available CEF.

Example request

Get a list of available CEF.

```
curl -k -u admin:changeme https://localhost/rest/cef -G -X GET
```

Example response

A successful GET will return back a JSON formatted list of key names and their contains data.

```
{
        "count": 151,
        "data": [
            {
                "type": "default",
                "id": 1,
                "data_type": [
                    "mac address"
                ],
                "name": "dmac"
            },
            â ¦
            {
                "type": "custom",
                "id": 566,
                "data_type": ["myIp"],
                "name": "myDestination"
            }
        ],
        "num_pages": 16
    }
```

POST

Add a custom CEF.

Example request
You can add a custom CEF by supplying a JSON formatted body.

```
curl -k -u admin:changeme https://localhost/rest/cef \
-d '{
        "name":"docs-test-cef",
        "data_type":["test"]
    }'
```

Example response
A successful response includes the numeric Id given to the CEF.

```
{
        "id": 151,
        "success": true
}
```

/rest/cef filters

You can use operators to sort or filter the results.

_filter_type	optional	string	Either "custom" or "default". Custom returns only custom CEFs, while default returns a list of all default CEFs.
_filter_name	optional	string	A quoted name for the CEF you want information about.

/rest/cef?_filter_type="custom"

Look up information about a CEF

Syntax

```
https://<username>:<password>@<host>/rest/cef?_filter_type="custom"
```

GET

Get a list of all custom CEFs defined on the system.

Example request
Get a list of available CEF.

```
curl -k -u admin:changeme https://localhost/rest/cef?_filter_type="custom" -G -X GET
```

Example response
A successful GET will return back a JSON formatted list of key names and their contains data.

```
{
    "count": 1,
    "data": [
        {
            "type": "custom",
            "id": 151,
            "data_type": [
                "test"
            ],
            "name": "docs-test-cef"
        }
    ],
    "num_pages": 1
}
```

/rest/cef?custom=true&page_size=3&page=0

Get a paginated list of all custom CEFs

Syntax

```
https://<username>:<password>@<host>/rest/cef?_filter_type="custom"
```

Usage

You can set the page size to any valid integer.

GET

Get a list of all custom CEFs defined on the system, organized into pages.

Example request

Get a list of available CEF.

```
curl -k -u admin:changeme https://localhost/rest/cef?custom=true&page_size=3&page=0 -G -X GET
```

Example response

A successful GET will return back a JSON formatted list of key names and their contains data.

```
{
    "count": 151,
    "data": [
        {
            "type": "default",
            "id": 1,
            "data_type": [
                "mac address"
            ],
            "name": "dmac"
        },
        {
            "type": "default",
            "id": 2,
            "data_type": [],
            "name": "act"
        },
        {
            "type": "default",
            "id": 3,
            "data_type": [
                "host name"
            ],
            "name": "dhost"
        },
        {
            "type": "default",
            "id": 4,
            "data_type": [],
            "name": "app"
        },
        {
            "type": "default",
            "id": 5,
            "data_type": [],
            "name": "deviceCustomDate2Label"
        },
        {
            "type": "default",
            "id": 6,
            "data_type": [],
            "name": "applicationProtocol"
        },
        {
            "type": "default",
            "id": 7,
            "data_type": [],
            "name": "deviceDirection"
        },
        {
            "type": "default",
            "id": 8,
            "data_type": [],
            "name": "baseEventCount"
        },
        {
            "type": "default",
            "id": 9,
            "data_type": [
                "domain"
            ],
            "name": "dntdom"
        },
        {
            "type": "default",
            "id": 10,
            "data_type": [],
            "name": "bytesIn"
        }
    ],
    "num_pages": 16
}
```

rest/cef?_filter_name="<name>"

Get information about a single CEF, by name.

Syntax

```
https://<username>:<password>@<host>rest/cef?_filter_name="<name>"
```

GET

Get information about a single CEF, by name.

Example request

Get a list of CEF called docs-test-cef.

```
curl -k -u admin:changeme https://localhost/rest/cef?_filter_name="docs-test-cef" -G -X GET
```

Example response

A successful GET will return back a JSON formatted list of key names and their contains data.

```
{
    "count": 1,
    "data": [
        {
            "type": "custom",
            "id": 151,
            "data_type": [
                "test"
            ],
            "name": "docs-test-cef"
        }
    ],
    "num_pages": 1
}
```

/rest/cef/<id>

Update a custom CEF.

Syntax

```
https://<username>:<password>@<host>/rest/cef/<id>
```

POST

Update a custom CEF by supplying the CEF Id and a JSON formatted body of the updates.

Example request

Update the custom CEF called docs-example-cef

```
curl -k -u admin:changeme https://localhost/rest/cef/<id> \
-d '{
        "name":"docs-example-cef",
        "data_type":["example"]
    }
'
```

Example response

A successful response includes the numeric Id given to the CEF.

```
{
        "id": 151,
        "success": true
}
```

DELETE

Delete a custom CEF by supplying the CEF Id.

Example request

Delete the custom CEF with Id 151.

```
curl -k -u admin:changeme https://localhost/rest/cef/151 -X DELETE
```

Example response

A successful response includes a success message.

```
{
        "success": true
}
```

Clustering endpoints

REST Cluster Nodes

Splunk SOAR (On-premises) cluster information can be read via an HTTP GET request. This endpoint is read-only and does not accept HTTP POST requests. Access to this endpoint requires authentication and system settings permissions.

/rest/cluster_node

Cluster information such as node name, status, Id, version, etc.

Syntax

```
https://<username>:<password>@<host>/rest/cluster_node
```

GET

Get cluster information.

Example curl request
Get cluster information.

```
curl -k -u admin:changeme https://localhost/rest/cluster_node -G -X GET
```

Example response
A successful GET will return a JSON formatted list of key names and data.

```
{
    "count": 3,
    "data": [
        {
            "update_time": "2018-09-04T22:27:42.722084Z",
            "name": "1.2.3.4",
            "enabled": true,
            "server": true,
            "host": "1.2.3.4",
            "create_time": "2018-09-04T19:50:27.960889Z",
            "version": "4.0.1068",
            "online": true,
            "guid": "6c395b0f-5574-45e2-ac74-801d0ad8495c",
            "id": 3
        },
        {
            "update_time": "2018-09-04T22:28:05.027246Z",
            "name": "2.3.4.5",
            "enabled": true,
            "server": true,
            "host": "2.3.4.5",
            "create_time": "2018-09-04T19:49:18.185595Z",
            "version": "4.0.1068",
            "online": true,
            "guid": "e9ba83af-be4e-4cb9-9264-32971ef80b55",
            "id": 2
        },
        {
            "update_time": "2018-09-04T22:28:06.517752Z",
            "name": "3.4.5.6",
            "enabled": true,
            "server": true,
            "host": "3.4.5.6",
            "create_time": "2018-09-04T19:48:03.988898Z",
            "version": "4.0.1068",
            "online": true,
            "guid": "41200aa3-1e9c-4436-b77f-d9471263572f",
            "id": 1
        }
    ],
    "num_pages": 1
}
```

Container endpoints

REST Containers

Create, modify, and manage containers using the following rest endpoints.

/rest/container

Manage containers and their associated authorized users, pin templates, comments, attachments, and options.

Syntax

```
https://<username>:<password>@<host>/rest/container
```
POST

Create a container.

Request parameters
Containers are modified with the following parameters.

Field	Required	Type	Description
artifacts	optional	JSON Array of Objects	Enables creation of artifacts in the same POST with the container. Must be a list of JSON objects containing the artifact data. See the REST documentation on artifacts regarding artifact format and the note below on run_automation for additional information.
asset_id	optional	integer	Id of the asset which produced the container.
close_time	optional	ISO 8601 formatted timestamp	Date and time (in UTC) when the container was closed.
custom_fields	optional	JSON Object	JSON objects contains key/value pairs for custom container fields. There may be required fields defined in the administration settings. See the Administrator's Guide for details.
data	optional	JSON Object	Custom data field.
description	optional	string	A brief useful description of the behavior tracked by this container.
due_time	optional	ISO 8601 formatted timestamp	Date and time (in UTC) when the SLA for this container will expire.
end_time	optional	ISO 8601 formatted timestamp	Date and time (in UTC) when the behavior tracked by the container stopped.
ingest_app_id	optional	integer	Id of the app which produced the container.
kill_chain	optional	string	Cyber kill chain. One of • Reconnaissance • Weaponization • Delivery • Exploitation • Installation

Field	Required	Type	Description
			• Command & Control • Actions on Objectives
label	required	string	The label classifies the container. This field is required, however if the label can be determined using the app or an automation user's default label, then it may be omitted. Specifying a label overrides the default. Some example labels: • events • incident • intelligence • vulnerability
name	required	string	A short friendly name for the container.
owner_id	optional	integer or string	Splunk SOAR (On-premises) account username or numeric ID. ID of the user who is the current owner of the container. You can use owner_id or role_id, but not both at the same time.
role_id	optional	integer or string	Splunk SOAR (On-premises) role name or numeric Id as per the /rest/role API. Id of the role that is the current owner of the container, if you don't know the username of the owner. You can use owner_id or role_id, but not both at the same time.
run_automation	optional	boolean	Not a container data field: This parameter instructs Splunk SOAR (On-premises) to run automation upon creation or update of the container, and defaults to False.
sensitivity	optional	string	Describes how sensitive material related to the container is. One of: • white • green • amber • red
severity	optional	string	Describes how severe material related to the container is. Helps to determine the SLA applied to `Actions` related to the container. Either one of **Low**, **Medium**, **High**, or a custom severity name created by an administrator.
source_data_identifier	optional	string	Id which can be used to find this container in the source product. (e.g. the container was retrieved from a SIEM, this is the Id in the SIEM) Must be unique per container, if no identifier is provided, a unique value is generated.
start_time	optional	ISO 8601 formatted timestamp	Date and time (in UTC) when the behavior tracked by the container started.
open_time	optional	ISO 8601 formatted timestamp	Date and time (in UTC) when the container was opened or reopened.
status	optional	string	Current status of the container. Either one of **New**, **Open**, **Closed**, or a custom status created by an administrator.
tags	optional	Array of strings	0 or more tags associated with the asset. A simple string can also be used for a single tag.
tenant_id	optional	integer or string	The Splunk SOAR (On-premises) tenant ID as per the /rest/tenant API. While this field is optional, it is required if multi-tenancy is enabled.
container_type	optional	string	The container type. Valid values are 'default' or 'case'. Containers with the 'default' type are events in the user interface.
template_id	optional	integer	The ID of the workbook to apply to this container. If provided, the phases and tasks from the specified workbook will be added to this container. The workbook ID can be found using the /rest/workbook_template API.

Field	Required	Type	Description
authorized_users	optional	list of integers	The Splunk SOAR (On-premises) user IDs as per the /rest/ph_user API, for example [1,5,6,7]. The authorized user list is always updated to include only the list passed by this command. Authorized access can only be granted to the subset of users who are already assigned to a label that has edit permissions on the container. Some teams only want to allow certain people to work on particular types of cases, so just because people are assigned to the label, it doesn't mean they all need access to a particular case. Admins always have access to all containers, so they don't need to be added to the authorized list.

To optimize performance when creating multiple artifacts, first create the container, then create all artifacts except the last with run_automation set to False, and then create the last artifact with run_automation set to True. This will cause automation such as active playbooks to run only after all artifacts have been added. To be even more efficient, you can create the container and artifacts in a single POST (see above) in which case you should not set run_automation at all, as Splunk SOAR automatically sets run_automation after the last artifact is created.

Example request
Create a container named "Example Container" with a label of "events" and assign it to an "owner."

```
curl -k -u admin:changeme https://localhost/rest/container \
-d '{
  "asset_id": 12,
  "artifacts": [ ],
  "custom_fields": {},
  "data": { },
  "description": "Useful description of this container.",
  "due_time": "2019-03-21T19:29:23.759Z",
  "label": "events",
  "name": "Example Container",
  "owner_id": "user@mail.example.com",
  "run_automation": false,
  "sensitivity": "red",
  "severity": "high",
  "source_data_identifier": "4",
  "status": "new",
  "start_time": "2019-03-21T19:28:13.759Z",
  "open_time": "2019-03-21T19:29:00.141Z",
  "tags": ["tag1", "tag2"],
  "container_type": "case"
}'
```

Success example response
A successful POST returns a success indicator and the Id of the newly created container.

```
{
    "id": 8,
    "success": true
}
```

Failed example response
Posting a JSON that matches the asset and source_data_identifier of an existing container results in a duplication error. The response also returns the Id of the matching container.

```
{
    "existing_container_id": 8,
    "failed": true,
    "message": "duplicate with source_data_identifier 1 and asset_id 4"
}
```

Example request

Create a container and assign a role instead of an owner.

```
curl -k -u admin:changeme https://localhost/rest/container \
-d '{
  "name": "Example Container",
  "label": "events",
  "template_id": 4,
  "status": "new",
  "role_id": "1",
  "severity": "medium",
  "sensitivity": "amber",
  "source_data_identifier": "4",
  "due_time": "2019-03-21T19:29:23.759Z",
  "container_type": "default"
}'
```

Success example response

A successful POST returns a success indicator and the Id of the newly created container.

```
{
  "new_artifact_ids": [],
  "id": 52,
  "success": true
}
```

POST

Update existing containers and revise the associated authorized user list.

Request parameters

See Create a container for the full parameters.

Example request

Update container IDs 44 and 45 and revise the associated authorized user list with the following parameters.

Field	Required	Type	Description
id	required	integer	ID of the container for which to set the authorized user list.
authorized_users	optional	list of integers	Splunk SOAR (On-premises) user IDs from the /rest/ph_user API, for example [1,5,6,7]. The authorized user list is always updated to include only the list passed by this command.

```
curl -k -u admin:changeme https://localhost/rest/container \
-d '[
  {
    "id": 44,
    "authorized_users": [8, 9]
  },
  {
    "id": 45,
    "authorized_users": [8, 9]
  }
]'
```

Example response

A successful POST returns a success indicator and the Id of the updated container.

```
[
  {
     "id": 44,
    "success": true
  },
  {
    "id": 45,
    "success": true
    }
]
```

Example request

Update container IDs 44 and 45 and clear the associated authorized user list. Leave the `authorized_users` list empty to clear it.

```
curl -k -u admin:changeme https://localhost/rest/container \
-d '[
  {
    "id": 44,
    "authorized_users": []
  },
  {
    "id": 45,
    "authorized_users": []
  }
]'
```

Example response

```
[
  {
    "id": 44,
    "success": true
  },
  {
    "id": 45,
    "success": true
  }
]
```

/rest/container optional query parameters

Query all containers for authorized users with the addition of a query parameter.

/rest/container?_annotation_authorized_users

The Container LIST API will also display the authorized field when the annotation is provided.

Syntax

```
https://<username>:<password>@<host>/rest/container?_annotation_authorized_users
```

GET

Get the authorized users.

Example request

Get the authorized users.

```
curl -k -u admin:changeme https://localhost/rest/container?_annotation_authorized_users -G -X GET
```

Example response

A successful GET will return back a JSON formatted list of key names and their data. If there are no users associated with an authorized user container list, then an empty list is returned. If the calling user is an admin or on the authorized user list, then this field is a list of user IDs.

```
[
   {
       ...
    "authorized_users": [],
   },
  {
     ...
     "authorized_users": [
      3,
      4
    ],
   }
]
```

/rest/container/<container_id>

Update an existing container.

Syntax

```
https://<username>:<password>@<host>/rest/container/<container_id>
```

POST

Update an existing container that is assigned to a user and assign a role instead.

Request parameters

See /rest/container for parameters.

Example request

Update container Id 51 and assign it to role Id 2.

```
curl -k -u admin:changeme https://localhost/rest/container/51 \
-d '{
        "status": "new",
        "id": 51,
        "role_id": "2"
}'
```

Example response

A successful POST returns a success indicator and the Id of the previously created container.

```
{
        "id": 51,
        "success": true
}
```

POST

Update an existing container and assign authorized users.

Request parameters
See Create a container for the full parameters.

Example request
Update container 51 and assign authorized users with the following parameters.

Field	Required	Type	Description
id	required	integer	ID of the container for which to set the authorized user list.
authorized_users	optional	list of integers	Splunk SOAR (On-premises) user IDs from the /rest/ph_user API, for example [1,5,6,7]. The authorized user list is always updated to include only the list passed by this command.

```
curl -k -u admin:changeme https://localhost/rest/container/51 \
-d '{
        "authorized_users": [ <user_id_1> , <user_id_2> ]
}
}'
```

Example response
A successful POST returns a success indicator and the Id of the updated container.

```
{
    "id": 2,
    "success": true
}
```

/rest/container/<container_id> optional query parameters

Query a container ID for authorized users.

/rest/container/<container_id>?_annotation_authorized_users

The container API has a query parameter that returns the authorized user field associated with the container.

Syntax

```
https://<username>:<password>@<host>/rest/container/<container_id>?_annotation_authorized_users
```

Usage details

GET

Retrieve a list of non-admin users who have sufficient permissions to view a container.

Example request
Returns the authorized user field associated with container ID 51.

```
curl -k -u admin:pass https://localhost/rest/container/51?_annotation_authorized_users -G -X GET
```

Example response
If there are no users associated with a container authorized user list, then an empty list is returned. If the calling user is an admin or on the authorized user list, then this field is a list of user IDs.

```
{
   ...
    "authorized_users": [
      3,
      4
    ],
}
```

/rest/container/<container_id>/permitted_users

Query for authorized user candidates.

Syntax

```
https://<username>:<password>@<host>/rest/container/<container_id>/permitted_user
```

Usage details
This is used to show a list of candidates who can be granted authorized user access to a container using the authorized_users parameter.

GET

Retrieve a list of non-admin users who have sufficient permissions to view a container.

Example request
Retrieve a list of non-admin users who have sufficient permissions to view a container.

```
curl -k -u admin:pass https://localhost/rest/container/<container_id>/permitted_users -G -X GET
```

Example response
The result is a JSON object.

```
{
        "users": [{
                        "username": "hansel",
                        "first_name": "hansel",
                        "last_name": "breadcrumbs",
                        "display_name": "hansel breadcrumbs",
                        "type": "normal",
                        "id": 3
                },
                {
                        "username": "gretel",
                        "first_name": "gretel",
                        "last_name": "breadcrumbs",
                        "display_name": "gretel breadcrumbs",
                        "type": "normal",
                        "id": 4
                }
        ]
}
```

/rest/container_pin_settings

Create a pin template.

Syntax

```
https://<username>:<password>@<host>/rest/container_pin_settings
```

Usage details

Pins are also called Heads Up Display (HUD) cards. These cards are displayed in Investigation. HUD cards are used to help analysts or other users easily track important information at a glance.

A new end point has been added to set a pin/HUD card that will be automatically created when a container matching the set label is created. Setting the label to null makes the pin/HUD card apply to all containers.

POST

Create a template pin/HUD card.

Request parameters

Create a template with the following parameters.

Field	Required	Type	Description
pin_type	required	string	The pin type of the pin. Can either be "preset metric" or "custom field". The types "data" and "manual card" cannot be used for a template.
label	required	string	The label of the card. When an event with this label is created, a pin with this template will also be created for that event.
preset_metric		string	The preset metric for the card. Available options are: • "remaining tasks" • "failed actions" • "failed playbooks" • "tasks exceeding sla" • "time to resolve" This option cannot be set if "custom_field" is used.
custom_field		string	The custom field for the card. The custom field has to exist prior to the REST API request. This option cannot be set if "preset_metric" is used.
pin_style	optional	string	The color of the HUD card. If not specified, defaults to grey. • grey • blue • red
order	optional	int	The position at which the new pin is created. Existing pins at this position or later have their positions shifted by 1.

Example request

Create a HUD card with the pin type of "preset metric," the label of "global," and the preset metric of "remaining tasks."

```
curl -k -u admin:changeme https://localhost/rest/container_note \
-d '{
        "pin_type": "preset metric",
        "preset_metric": "remaining tasks",
        "label": "global",
        "pin_style": "grey",
        "order": 1
}'
```

Example response

A successful POST returns a success indicator and the Id of the newly created pin.

```
{
    "id": 17,
    "success": true,
    "message": "Created a new pin"
}
```

/rest/container_pin

Add a Pin to a container.

Syntax

```
https://<username>:<password>@<host>/rest/container_pin
```

POST

Pin data to the HUD tab in Investigation.

Request parameters

Pin data to the HUD tab in Investigation with the following parameters.

Field	Required	Type	Description
container_id	required	integer	Id of the container to pin to.
order	required	integer	The position at which the new pin is created. Existing pins at this position or later have their positions shifted by 1.
message	required	string	The pin message.
data	required	string	The pin data.
pin_type	optional	string	Defaults to "data". There are four valid "pin_type" values: • "data" • "manual card" • "custom field" • "preset metric" If you use "data", you have the option to add a "data" field to your request. If you use "manual card", "input type" becomes a required field.
pin_style	optional	string	Valid values are: • "grey" • "blue" • "red"

Field	Required	Type	Description
			Defaults to grey if no value is set.
name	optional	string	Defaults to null. Names can be used to easily reference pins without needing to know its Id. A name is unique per container. If a pin is creating using a name which already exists on that container, the existing named pin is updated instead of a new one being created. There can be any number of unnamed pins on a container.
input_type	dependent	string	Valid options are "text" or "select".
input_choices	dependent	JSON list	If the "input type" is set to "select", provide a list of the choices. Example: ["one", "two", "three"]
custom_field	dependent	string	If the "pin_type" is set to "custom_field" provide the name of the custom field.
preset_metric	dependent	string	If the "pin_type" is set to "preset_metric" set the metric to use. Options are: • "remaining tasks" • "failed actions" • "failed playbooks" • "tasks exceeding sla" • "time to resolve"

Example request

Pin data to container Id 12 with a pin type of "manual card," a message, and the data.

```
curl -k -u admin:changeme https://localhost/rest/container_pin \
-d '{
        "container_id": 12,
        "message": "This is a malicious IP",
        "data": "192.168.58.130",
        "pin_type": "manual card",
        "input_type": "select",
        "input_choices": ["one", "two", "three"],
        "pin_style": "grey",
        "name": null
}'
```

Example response

A successful POST returns a message, "Created a new pin", the content of the newly created pin, the id of the pin, and a success indicator.

```
{
        "message": "Created a new pin",
        "data": {
                "pin_type": "manual card",
                "preset_metric": null,
                "container": 12,
                "name": null,
                "author": 1,
                "input_type": "select",
                "custom_field": null,
                "modified_time": "2019-05-31T18:37:49.191094Z",
                "order": 5,
                "create_time": "2019-05-31T18:37:49.182561Z",
                "playbook": "",
                "input_choices": [
                        "one",
                        "two",
                        "three"
                ],
                "message": "This is a malicious IP",
                "data": "192.168.58.130",
                "id": 5,
                "pin_style": "grey"
        },
        "id": 5,
        "success": true
}
```

/rest/container_comment

Add a comment to a container.

Syntax

```
https://<username>:<password>@<host>/rest/container_comment
```

Usage details

The user adding the comment must have Edit permissions to the container.

POST

Add a comment to a container.

Request parameters

The request string contains the following parameters.

Field	Required	Type	Description
container_id	required	integer	Id of the container to comment on.
comment	required	string	The contents of the comment.

Example request

Comment on container Id 12.

```
curl -k -u admin:changeme https://localhost/rest/container_comment \
-d '{
        "container_id": 12,
        "comment": "This is an interesting container"
}'
```

The comment is "from" whichever user is authenticating the REST POST. If POSTing as an automation user, the automation user's name shows up as the commenter.

Example response

A successful POST returns a success indicator and the ID of the newly created comment.

```
{
        "id": 27,
        "success": true
}
```

/rest/container_attachment

Attach a file to a container.

Syntax

```
https://<username>:<password>@<host>/rest/container_attachment
```

Usage details

The user adding the attachment must have Edit permissions to the container.

POST

Add an attachment to a container.

Request parameters

The request string contains the following parameters.

Field	Required	Type	Description
container_id	required	integer	Id of the container to add the attachment to.
file_name	required	string	The name of the file being uploaded.
file_content	required	string	The contents of the file. This is JSON serialized. Binary files must be base 64 encoded.
metadata	optional	JSON Object	This is a space for users to attach information to the file. A key called "contains" should be passed in with this with the value being a JSON Array of contains types. At a minimum, "vault id" should be submitted as a contains. While this is optional, not passing this value limits the ability to take actions on this file from the UI.

Example request

Add an attachment to container Id 12.

```
curl -k -u admin:changeme https://localhost/rest/container_attachment \
-d '{
        "container_id": 12,
        "file_content": "JSON serialized contents of a file",
        "file_name": "my_test_upload.txt",
        "metadata": {
                "contains": [
                        "vault id"
                ]
        }
}'
```

Example python script

```
import json
import requests
from base64 import b64encode

contents = open('/path/to/binary.dat', 'rb').read()

serialized_contents = b64encode(contents)

attachment_json = {
    'container_id': 12,
    'file_content': serialized_contents,
    'file_name': 'binary.dat',
    'metadata': {
        'contains': [
            'vault id'
        ]
    }
}

AUTH_TOKEN = 'AUTH TOKEN GOES HERE'

requests.post('https://phantom.example.com/rest/container_attachment',
                auth=('', AUTH_TOKEN), data=json.dumps(attachment_json))
```

The webserver used, NGINX, is configured to reject POSTs larger than 32MB. If you wish to be able to upload files larger than this, the configuration must be modified. However, please be aware of the security and performance implications of changing the maximum request body size.

Example response
A successful POST returns a success indicator and the Id of the newly created attachment.

```
{
        "container": 159,
        "hash": "a97e90e672b90ff092c674d709803aeb036e16c9",
        "message": "success",
        "size": 54,
        "succeeded": true,
        "vault_id": "a97e90e672b90ff092c674d709803aeb036e16c9"
}
```

/rest/container_options

Get container options.

Syntax

```
https://<username>:<password>@<host>/rest/container_options
```

Usage details
You need the "Container View" permission to get this list.

GET

Get a list of options for containers.

Example request
Retrieve a list of non-admin users who have sufficient permissions to view a container.

```
curl -k -u admin:pass https://localhost/rest/container_options -G -X GET
```

Example response
The result is a JSON object.

```
{
        "status": [{
                        "is_default": true,
                        "name": "new",
                        "status_type": "new",
                        "order": 0
                },
                {
                        "is_default": false,
                        "name": "ripe",
                        "status_type": "new",
                        "order": 1
                },
                {
                        "is_default": true,
                        "name": "open",
                        "status_type": "open",
                        "order": 0
                },
                {
                        "is_default": true,
                        "name": "closed",
                        "status_type": "resolved",
                        "order": 0
                }
        ],
        "severity": [{
                        "color": "red",
                        "is_default": false,
                        "name": "high",
                        "order": 0
                },
                {
                        "color": "yellow",
                        "is_default": true,
                        "name": "medium",
                        "order": 1
                },
                {
                        "color": "green",
                        "is_default": false,
                        "name": "low",
                        "order": 2
                }
        ],
        "tags": [
                "tag4",
                "tag5",
                "tag1",
                "tag2",
                "tags",
                "auto add tag test",
                "tag3",
                "test123qa2",
                "test123qa"
        ],
        "sensitivity": [
                [
                        "red",
                        "TLP:Red"
                ],
                [
                        "amber",
                        "TLP:Amber"
                ],
                [
                        "green",
                        "TLP:Green"
                ],
                [
                        "white",
                        "TLP:White"
                ]
        ],
        "label": [
                "events"
        ],
        "severity_colors": [
                [
                        "red",
                        "Red"
                ],
                [
                        "orange",
                        "Orange"
                ],
                [
                        "yellow",
                        "Yellow"
                ],
                [
                        "green",
                        "Green"
                ],
                [
                        "light_blue",
                        "Light Blue"
                ],
                [
                        "blue",
                        "Blue"
                ],
                [
                        "purple",
                        "Purple"
                ],
                [
                        "pink",
                        "Pink"
                ],
                [
                        "light_grey",
                        "Light Grey"
                ],
                [
                        "dark_grey",
                        "Dark Grey"
                ]
        ]
}
```

/rest/container/:id/approvals

Get a list of pending container approvals.

Syntax

```
https://<username>:<password>@<host>/rest/container/:id/approvals
```

GET

Get a list of pending container approvals

Example request

```
curl -k -u admin:pass https://localhost/rest/container/:id/approvals -G -X GET
```

Example response

```
{
    "count": 1,
    "data": [
        {
            "id": 1,
            "_pretty_action_run": "user initiated post ip action",
            "action_run": 1,
            "_pretty_asset": "abuseipdb",
            "asset": 16,
            "close_time": null,
            "display": true,
            "_pretty_due_time": "In 57 minutes",
            "due_time": "2020-08-14T02:06:22.543000Z",
            "_pretty_escalated_approval": "",
            "escalated_approval": null,
            "message": "pending-approval",
            "name": "approval for 'post ip' on asset 'abuseipdb'",
            "_pretty_owner": "admin",
            "owner": 5,
            "owner_type": "Primary Asset Owner",
            "_pretty_parent": "",
            "parent": null,
            "playbook_run": null,
            "_pretty_start_time": "0 minutes ago",
            "start_time": "2020-08-14T01:09:01.556000Z",
            "status": "pending",
            "type": "asset",
            "version": 1,
            "jitc": {},
            "node_guid": "6e17377a-313e-4a96-8c2f-2a3e16f6ec45",
            "responses": [],
            "_pretty_container": 4,
            "_pretty_action_name": "post ip"
        }
    ],
    "pages": 1
}
```

Custom function endpoints

REST Custom Function

/rest/custom_function

`https://<username>:<password>@<host>/rest/custom_function`

Create or view a list of all custom functions.

POST

Create a custom function.

Request parameters

Field	Required	Type	Description
name	required	string	A unique name per repository that identifies the custom function.
scm_id	required	string	The ID for an existing repository on the system.

<Returned values>

None.

Fields for both creating and updating a custom function

Field	Required	Type	Default	Description
draft_mode	optional	Boolean	true	A flag to mark a custom function as a draft version. This flag allows you to save invalid Python code while `draft_mode` is set to `true`.
python	optional	string	true	Python code that executes for the custom function when it is included in a playbook.
description	optional	string	true	Descriptive text for the custom function. This field displays when selecting custom functions inside the playbook editor.
commit_message	Dependency of a Python field	string	true	This field is required if the Python field is passed. This message is used for the commit of the changed Python and metadata files to the connected repository.

inputs	optional	JSON object	{}	Inputs are used for configuring the custom function in the playbook editor.
				
				Example value:
				
				`[ { "description":`
				`"fgh",`
				`"contains_type"`
				`:  ["ip"] ,`
				`"name":  "fgh"`
				`} ]`

outputs	optional	JSON object	{}	Outputs are used for configuring downstream blocks from the custom function in the playbook editor.

Field	Required	Type	Default	Description

				Example value:
				`[ {`
				`"data_path":`
				`"ip",`
				`"description":`
				`"This is an ip",`
				`"contains_type"`
				`:  ["ip"],`
				`} ]`

Example request and response

This example JSON request shows creating the draft version of a non-draft custom function, and giving that draft version a different description.This request doesn't disable the current custom function, so it doesn't affect any playbooks using this custom function.

JSON request

```
curl -k -u admin:changeme https://localhost/rest/custom_function/1 \
-d '{
    "description": "Example description. Can be any text.",
    "draft_mode": true
}'
```

JSON response

```
{
    "warnings": [ ] ,
    "errors":  [ ] ,
    '"id":  2,
    "success":  true,
    "commit_sha":  "1ded5f50ac2e2790b3e2869f6e1487ffc4236856"
}
```

The ID passed back in the response is the newly created custom function. `commit_sha` is the commit for the new files.

GET

View a list of custom functions.

<Returned values>

None.

Notable parameters

Field	Required	Type	Description
commit_sha	optional	string	The most recent GIT commit identifier for the custom function.
disabled	optional	Boolean	Describes whether the custom function is active or inactive.
draft_mode	optional	Boolean	If you set the `draft_mode` field to `true`, you can see your draft mode versions of custom functions. If you set the `draft_mode` field to `false`, you can see your non-draft mode custom functions.

Field	Required	Type	Description
latest_editor	optional	integer	A foreign key to a PhUser model. This field reflects the last user that edited the custom function.
name	optional	string	The name of the custom function. This name is the same as the name on the custom function listing page.
scm	optional	integer	A foreign key to the SCM model. This field is the current repository where the custom function is saved.

Example request and response
Request to view the list of custom functions.

```
curl -k -u admin:changeme https://localhost/rest/custom_function/<id> -G -X GET
```

JSON response

```
{
    "count": 1,
    "data": [
        {
            "scm": 1,
            "platform_version": "4.8",
            "description": "This is a description",
            "latest_editor": 1,
            "json_blob_sha": "c7299d2825c823d24d2570bed7f2321bee7113ef",
            "draft_mode": false,
            "forked_from": null,
            "commit_sha": "7e08d23bd11bc3043e0872a58e8ead0722edc3b3",
            "python_blob_sha": "37422d6dc6e7fee1b05b2a9fc58edceb0a0d069f",
            "scm_user_name": "",
            "last_updated_time": "2020-01-01T01:00:00Z",
            "version": 1,
            "passed_validation": true,
            "date_created": "2020-01-01T01:00:00Z",
            "disabled": false,
            "id": 1,
            "custom_function_id": "930b704e669274d0fb9293d1db5bdc5f0c457304",
            "name": "example_custom_function"
        }
    ],
    "num_pages": 1
}
```

rest/custom_function/<id>

```
https://<username>:<password>@<host>/rest/custom_function/<id>
```

Update a custom function, or view a list of all custom functions.

POST

Update a custom function.

You can't update the `name` or `scm_id` fields of an existing custom function. Because you can't update the fields, when you make a POST request, make sure to either match the `name` and `scm_id` fields in the request body, or don't include the `name` and `scm_id` fields in the request body in order for it to pass. If you want to change the `name` and `scm_id` fields of the custom function, you need to create a copy.

<Returned values>

None.

Fields for both creating and updating a custom function

Field	Required	Type	Default	Description
draft_mode	optional	Boolean	true	A flag to mark a custom function as a draft version. This flag allows you to save invalid Python code while `draft_mode` is set to `true`.
python	optional	string	true	Python code that runs for the custom function when it is included in a playbook.
description	optional	string	true	Descriptive text for the custom function. This field is shown when selecting custom functions inside the playbook editor.
commit_message	Dependency of a Python field	string	true	This field is required if the Python field is passed. This message is used for the commit of the changed Python and metadata files to the connected repository.

inputs	optional	JSON object	{}	Inputs are used for configuring the custom function in the playbook editor.
				Example value:
				`[ { "description:`
				`"fgh",`
				`"contains_type"`
				`:  ["ip"] ,`
				`"name":  "fgh"`
				`} ]`

outputs	optional	JSON object	{}	Outputs are used for configuring downstream blocks from the custom function in the playbook editor.
				Example value:
				`[ {`
				`"data_path":`
				`"ip",`
				`"description":`
				`"This is an ip",`
				`"contains_type"`
				`:  ["ip"],`
				`} ]`

Example request and response

This example JSON request shows creating the draft version of a non-draft custom function, and giving that draft version a different description.This request doesn't disable the current custom function, so it doesn't affect any playbooks using this custom function.

JSON request

```
curl -k -u admin:changeme https://localhost/rest/custom_function/1 \
-d '{
    "description": "Example description. Can be any text.",
    "draft_mode": true
}'
```

JSON response

```
{
    "warnings": [ ] ,
    "errors":  [ ] ,
    '"id":  2,
    "success":  true,
    "commit_sha":  "1ded5f50ac2e2790b3e2869f6e1487ffc4236856"
}
```

The ID passed back in the response is the newly created custom function. `commit_sha` is the commit for the new files.

GET

View a single custom function or a list of custom functions.

<Returned values>

None.

Notable parameters

Field	Required	Type	Description
commit_sha	optional	string	The most recent GIT commit identifier for the custom function.
disabled	optional	Boolean	Describes whether the custom function is active or inactive.
draft_mode	optional	Boolean	If you set the `draft_mode` field to `true`, you can see your draft mode versions of custom functions. If you set the `draft_mode` field to `false`, you can see your non-draft mode custom functions.
latest_editor	optional	integer	A foreign key to a PhUser model. This foreign key reflects the last user that edited the custom function.
name	optional	string	The name of the custom function. This name is the same as the name in the custom function listing page.
scm	optional	integer	A foreign key to the SCM model. This foreign key is the current repository where the custom function is saved.

Example request and response

JSON request

```
curl -k -u admin:changeme https://localhost/rest/custom_function/<id> -G -X GET
```

Request to view a single custom function.

JSON response

```
{
    "playbooks": [
        {
            "active": true,
            "draft_mode": false,
            "name": "toggle_playbook_active",
            "id": 1
        }
    ],
    "disabled": false,
    "scm_user_name": "",
    "create_time": "2020-01-01T01:00:00Z",
    "id": 1,
    "scm_id": 1,
    "latest_editor": 1,
    "warnings": [],
    "commit_sha": "7e08d23bd11bc3043e0872a58e8ead0722edc3b3",
    "platform_version": "4.8",
    "version": 10,
    "passed_validation": true,
    "inputs": [
        {
            "custom_function": 10,
            "description": "",
            "contains_type": [
                "*"
            ],
            "id": 10,
            "name": "input_1"
        }
    ],
    "description": "This is a description",
    "draft_mode": false,
    "python": "def example_custom_function(input_1=None, **kwargs): return {} ",
    "outputs": [],
    "errors": [],
    "custom_function_id": "930b704e669274d0fb9293d1db5bdc5f0c457304",
    "name": "example_custom_function",
    "json_blob_sha": "c7299d2825c823d24d2570bed7f2321bee7113ef",
    "forked_from": null,
    "python_blob_sha": "37422d6dc6e7fee1b05b2a9fc58edceb0a0d069f",
    "date_created": "2020-01-01T01:00:00Z"
}
```

Evidence endpoints

REST Evidence

Evidence endpoint for managing evidence in indicators, events, or cases. This endpoint supports creating or deleting evidence objects.

/rest/evidence

Manage evidence.

Syntax

```
https://<username>:<password>@<host>/rest/evidence
```

POST

Add evidence to a container.

The body of the request is a JSON object with the following fields.

Field	Required	Description
container_id	required	Id of the container to which you are adding evidence.
object_id	required	Id of object to be added -- artifact id, note id, etc.
content_type	required	The content type of the object to add as evidence. One of the types: • containerattachment • artifact • actionrun • container • note

Example request
Add an artifact of Id 17 to container Id 9.

```
curl -k -u admin:changeme https://localhost/rest/evidence \
-d '{
    "container_id": 9,
    "object_id": 17,
    "content_type": "artifact",
}'
```

Example response
A successful POST will return a success indicator and the Id of the newly created evidence.

```
{
    "id": 4,
    "success": true
}
```

/rest/evidence optional query parameters and filters

Query all evidence with the addition of query parameters and filters. See REST Query Data for further information about query parameters.

/rest/evidence?&_special_content_type=True&_filter_container=<container id>

Example query parameters and filters follow:

- `_special_content_type=True` - adds a string to the JSON returned that gives the type of evidence. In the following example, the type is "artifact".
- `_filter_container=<container id>` - filters the query to a single container.

For cases or containers that have large collections of evidence, consider adding paging parameters.

Syntax

```
https://<username>:<password>@<host>/rest/evidence?<parameter>&_<filter>=<container id>
```
GET

Get a list of evidence.

Example request
Get a list of evidence for container Id 5 with a page size of 5 and special content type true.

```
curl -k -u admin:changeme
https://localhost/rest/evidence?page_size=5&_special_content_type=True&_filter_container=5 -G -X GET
```
Example response
A successful GET will return back a JSON formatted list of key names and data.

```
{
    "count": 1,
    "data": [
        {
            "container": 5,
            "modified_time": "2019-05-23T17:18:39.595829Z",
            "_special_content_type": "artifact",
            "create_time": "2019-05-23T17:18:39.595528Z",
            "object_id": 20,
            "content_type": 52,
            "id": 1
        }
    ],
    "num_pages": 1
}
```
/rest/evidence?_special_content_object&_filter_content_type_model="note"&search=[]&_annotation_container_attachme

Return information about a note's attachments when it is marked as evidence.

GET

Return information about the note's attachments.

Example request
Return information about the note's attachments.

```
/rest/evidence?_special_content_object&_filter_content_type_model="note"&search=[]&_annotation
_container_attachments=true
```

Example response
A successful GET provides information about the size, name, and container attachment ID of the note.

```
{
'id': 605,
'create_time': '2020-04-14T02:26:41.993230Z',
'modified_time': '2020-04-14T02:26:41.993413Z',
'container': 9738,
'object_id': 5203,
'content_type': 96,
'_special_content_object':
{
'id': 5203,
 'title': '',
 'content': 'note_content_fosecdYXmfSG',
 'create_time': '2020-04-14T02:26:41.297248Z',
 'modified_time': '2020-04-14T02:26:41.938770Z',
 'note_type': 'g',
 'author': 1640,
 'phase': None,
 'container': 9739,
 'task': None,
 'artifact': None,
 'container_attachments':
[{'name': 'container_attachment_ciYQzbVNGnjr', 'size': 0, 'container_attachment_id': 1514},
{'name': 'container_attachment_YaLEONTFIpSj', 'size': 0, 'container_attachment_id': 1515},
{'name': 'container_attachment_BwdoyghERKcn', 'size': 0, 'container_attachment_id': 1516}]
}
}
```

/rest/evidence/<evidence id>

Manage one item by evidence Id.

DELETE

Delete an item from evidence.

Example request
Delete the evidence with Id 5.

```
curl -k -u admin:changeme https://localhost/rest/evidence/5 -X DELETE
```

Example response
A successful response includes a success message.

```
{
    "id": 5,
    "success": true
}
```

HUD endpoints

REST HUD

The Heads Up Display in Investigation can be managed by using the REST API.

The relevant APIs are:

- `/rest/container_pin_settings`
- `/rest/container_pin`

See Containers for further details about pinning container data to the HUD.

Indicator endpoints

REST Indicators

All indicator REST endpoints are accessed with the HTTP GET method.

There are two parameters that can be used with all indicator endpoints:

Parameter: `tenant_id`

- default: (multi-tenancy enabled) - none, tenant_id MUST be included in all requests
- default: (multi-tenancy disabled) - system tenant id (0), all the examples that follow are with multi-tenancy disabled

Parameter: `timerange`

- default: last_30_days
- options:
    - today
    - yesterday
    - this_week
    - this_month
    - this_year
    - last_7_days
    - last_30_days
    - last_week
    - last_month
    - last_year

When the indicators feature is disabled in `/rest/system_settings/features`, the response for all indicator endpoints is the HTTP 400 status and "The indicators feature is not enabled." message. See /rest/system_settings/features.

/rest/indicator_stats_indicator_count

Indicator counts

Syntax

```
https://<username>:<password>@<host>/rest/indicator_stats_indicator_count
```
GET

Returns how many unique indicators and total indicators are present.

Example request
Return the count of unique indicators and total indicators.

```
curl -k -u admin:changeme https://localhost/rest/indicator_stats_indicator_count -G -X GET
```

Example response
A successful GET will return the count.

```
{
    "indicator_count": 182,
    "indicator_instance_count": 4107
}
```

Example response
Example response with no data.

```
{
    "indicator_count": 0,
    "indicator_instance_count": 0
}
```

/rest/indicator_stats_top_labels

Top event labels

Syntax

```
https://<username>:<password>@<host>/rest/indicator_stats_top_labels
```

GET

Returns the top three event (container) labels that have the most indicators, as well as the count of indicators within each label.

Example request
Return the top three event labels.

```
curl -k -u admin:changeme https://localhost/rest/indicator_stats_top_labels -G -X GET
```

Example response
A successful GET will return the top three event labels and the count of events per label.

```
{
    "topLabels": [
        {
            "count": 3981,
            "label": "events"
        },
        {
            "count": 52,
            "label": "campaign"
        },
        {
            "count": 34,
            "label": "generator"
        }
    ]
}
```

Example response
Example response with no data.

```
{
    "topLabels": []
}
```

/rest/indicator_stats_top_types

Top indicator types

Syntax

```
https://<username>:<password>@<host>/rest/indicator_stats_top_types
```

GET

Returns the top three indicator types that have the most indicators, as well as the count of indicators within each type.

Example request
Return the top three indicator types.

```
curl -k -u admin:changeme https://localhost/rest/indicator_stats_top_types -G -X GET
```

Example response
A successful GET will return the top three indicator types and the count of events per type.

```
{
    "topTypes": [
        {
            "count": 1031,
            "type": "port"
        },
        {
            "count": 1012,
            "type": "ip"
        },
        {
            "count": 453,
            "type": "hash"
        }
    ]
}
```

Example response
Example response with no data.

```
{
    "topTypes": []
}
```

/rest/indicator_stats_top_values

Top indicator values

Syntax

```
https://<username>:<password>@<host>/rest/indicator_stats_top_values
```

GET

Returns the top 3 indicator values that appear the most as well as the number of times they appear.

Example request
Return the top three indicator values.

```
curl -k -u admin:changeme https://localhost/rest/indicator_stats_top_values -G -X GET
```

Example response
A successful GET will return the top three indicator values and the number of times they appear.

```
{
    "topIndicators": [
        {
            "count": 509,
            "value": "80"
        },
        {
            "count": 504,
            "value": "4286"
        },
        {
            "count": 492,
            "value": "120"
        }
    ]
}
```

Example response
Example response with no data.

```
{
    "topIndicators": []
}
```

/rest/indicator

Multiple indicator data

Syntax

```
https://<username>:<password>@<host>/rest/indicator
```

GET

Returns multiple indicators, with the ability to filter based on specific fields or time range.

Example request
Example without any query parameters.

```
curl -k -u admin:changeme https://localhost/rest/indicator  -G -X GET
```

Example response
A successful GET without any query parameters returns the count, data, and number of pages.

```
{
        "count": 0,
        "data": [],
        "num_pages": 0
}
```

Example request

Example using query parameters to returns multiple indicators. This example uses the following parameters:

required parameters	• _special_fields • _special_labels • _special_contains • pretty
optional parameters	• sort ♦ values: _pretty_earliest_time, _pretty_latest_time, severity, id, value • page_size • page • order=desc • filter

```
curl -k -u admin:changeme
https://localhost/rest/indicator?_special_fields=true&_special_labels=true&_special_contains=true&
_special_severity=true&pretty=true&page_size=10&page=0&sort=id&order=desc&timerange=last_30_days -G -X GET
```

Example response

A successful GET returns multiple indicators, with the output pretty and sorted.

```
{
    "count": 182,
    "data": [
        {
            "_special_labels": [
                "events"
            ],
            "total_events": 1,
            "_pretty_tenant": "_default_",
            "tags": [],
            "_special_fields": [
                "fileHash"
            ],
            "value": "00b3199ed932f084f50b6a4df9124132",
            "_pretty_latest_time": "0 minutes ago",
            "latest_time": "2019-05-07T19:09:29.778556Z",
            "_pretty_earliest_time": "0 minutes ago",
            "open_events": 1,
            "_special_contains": [
                "hash"
            ],
            "severity_counts": [
                {
                    "count": 1,
                    "name": "high"
                }
            ],
            "earliest_time": "2019-05-07T19:09:29.778556Z",
            "id": 323,
            "tenant": 0
        }
    ],
    "num_pages": 182
}
```

/rest/indicator/<indicator_id>

Single indicator data

Syntax

```
https://<username>:<password>@<host>/rest/indicator/<indicator_id>
```

OR

```
https://<username>:<password>@<host>/rest/indicator_by_value?indicator_value=<indicator_value>
```

GET

Returns details of the specific indicator by its indicator Id.

Example request

Get the details of indicator Id 6. This example uses the following parameters:

optional parameters	• _special_contains • _special_fields • _special_labels

```
curl -k -u admin:changeme
https://localhost/rest/indicator/6?_special_fields=true&_special_labels=true&_special_contains=true  -G -X
GET
```

Example response

A successful GET returns the details of indicator Id 6.

```
{
    "_special_contains": [
        "port"
    ],
    "_special_fields": [
        "destinationPort"
    ],
    "_special_labels": [
        "events",
        "generator",
        "campaign"
    ],
    "earliest_time": "2018-07-24T05:07:38.589945Z",
    "id": 6,
    "latest_time": "2018-07-24T05:26:34.516600Z",
    "open_events": 347,
    "sc_high": 299,
    "sc_low": 100,
    "sc_medium": 110,
    "tags": [],
    "tenant": 0,
    "total_events": 509,
    "value": "80"
}
```

/rest/indicator_artifact

Artifacts by indicator

Syntax

```
https://<username>:<password>@<host>/rest/indicator_artifact
```

GET

Returns all artifacts that share a specific indicator id or value.

Example request

Get the artifacts with indicator Id 159. This example uses the following parameters:

required parameters	• indicator_id OR • indicator_value
optional parameters	• sort • page_size • page • order=desc • filter

```
curl -k -u admin:changeme
https://localhost/rest/indicator_artifact?indicator_id=159&_special_fields=true&_special_labels=true&
_special_contains=true&_special_severity=true&page=0&page
_size=10&sort=id&order=desc&timerange=last_30_days  -G -X GET
```

Example response

A successful GET returns the details of indicator Id 159 for the last 30 days.

```
{
    "count": 509,
    "data": [
        {
            "cef": {
                "baseEventCount": "120",
                "destinationPort": "80",
                "sourceAddress": "208.98.63.226",
                "sourcePort": "4286"
            },
            "cef_types": {},
            "container": 265,
            "container_name": "Zeus Infection on 10.17.1.201",
            "create_time": "2018-07-24T05:26:34.516600Z",
            "description": null,
            "end_time": "2018-07-24T05:26:33.742265Z",
            "has_note": false,
            "hash": "e7943930260daf75c2846e0a5a9aa2d8",
            "id": 730,
            "in_case": false,
            "ingest_app": null,
            "kill_chain": null,
            "label": "event",
            "name": null,
            "owner": null,
            "parent_artifact": null,
            "parent_container": null,
            "playbook_run": null,
            "severity": "low",
            "source_data_identifier": "qqHS6NWIAQLgVUVz85cNCY7WcB",
            "start_time": "2018-07-24T05:26:33.741621Z",
            "tags": [],
            "type": "network",
            "update_time": "2018-07-24T05:26:34.516648Z",
            "version": 1
        }
    ],
    "num_pages": 509
}
```

/rest/indicator_artifact_timeline

Indicator timeline by value

Syntax

```
https://<username>:<password>@<host>/rest/indicator_artifact_timeline
```

GET

Returns the timeline for a given indicator value.

Example request

Get the timeline for the indicator value of povvvodos.org. This example uses the following parameters:

required parameters	• indicator_value • timeline_width • timerange
optional parameters	margin_size

```
curl -k -u admin:changeme
https://localhost/rest/indicator_artifact_timeline?indicator_value=povvvodos.org&timeline
_width=951&timerange=last_30_days&margin_size=35 -G -X GET
```

Example response

A successful GET returns the details of indicator value povvvodos.org for the last 30 days.

```
{
    "data": [
        {
            "containerId": null,
            "count": 510,
            "didInjectThisDataPoint": true,
            "eventName": null,
            "id": null,
            "readableTime": "6/22",
            "severity": null,
            "time": "2018-06-22T19:00:11.430074Z",
            "type": "node",
            "unixTime": 1529694011.430074
        },
        {
            "count": 510,
            "nodeCount": 509,
            "nodeSeverityCount": {
                "high": 299,
                "low": 100,
                "medium": 110
            },
            "readableTime": "7/23",
            "time": "2018-07-23T03:59:59.363828Z",
            "type": "group",
            "unixTime": 1532318399.3638277
        },
        {
            "containerId": null,
            "count": 510,
            "didInjectThisDataPoint": true,
            "eventName": null,
            "id": null,
            "readableTime": "7/27",
            "severity": null,
            "time": "2018-07-27T04:59:47.569926Z",
            "type": "node",
            "unixTime": 1532667587.569926
        }
    ],
    "domain": [
        1529884800,
        1532476799
    ]
}
```

/rest/indicator_common_container

Containers by indicator

Syntax

```
https://<username>:<password>@<host>/rest/indicator_common_container
```
GET

Returns every event (container) that the specified indicators appear inside of.

Example request
Get the events for the indicator Ids of 1 and 2. This example uses the following parameters:

required parameters	indicator_ids (comma-separated string of indicator ids), for example: '1,2,3'

```
curl -k -u admin:changeme https://localhost/rest/indicator_common_container?indicator_ids=1,2 -G -X GET
```

Example response
A successful GET returns the details of indicator Ids 1 and 2.

```
[
    {
        "container_id": 81,
        "container_name": "koov4rra2ree5p6fzt1bcnjusu"
    },
    {
        "container_id": 61,
        "container_name": "6r8cm1798gq5hrb8e8t3ujw01m"
    }
]
```

List endpoints

REST Lists

Manage lists using the Splunk SOAR (On-premises) REST API. Lists are stored in a single row JSON blob object in the database and can contain up to 2 GB of data. Use lists for for slow or low data changes or static data lookups, rather than for large transactional data uses.

If a list's size exceeds 2 GB, data truncates without an error message.

/rest/decided_list

Manage lists.

Syntax

```
https://<username>:<password>@<host>/rest/decided_list
```

POST

Create a list

The body of the request is a JSON object with the following fields.

Field	Required/Optional	Field type	Description
content	Required	JavaScript array	A two dimensional array (array containing arrays containing simple values) which make the contents of your list. If the contents is a single array of values, it's wrapped in another array to create a two dimensional array.
name	Required	String	The name of the list. The name must be unique.

Example request

```
{
    "content": [
        [
            "1.1.1.1"
        ],
        [
            "1.1.1.2"
        ],
        [
            "1.1.1.3"
        ],
        [
            "1.1.1.4"
        ],
        [
            "1.1.1.5"
        ],
        [
            "1.1.1.6"
        ],
        [
            "1.1.1.7"
        ],
        [
            "1.1.1.8"
        ],
        [
            "1.1.1.9"
        ],
        [
            "1.1.1.10"
        ],
        [
            "1.1.1.11"
        ],
        [
            "2.2.2.12"
        ]
    ],
    "name": "My IP address list"
}
```

Example response

A successful POST returns a success indicator and the ID of the newly created list.

```
{
    "id": 41,
    "success": true
}
```

/rest/decided_list/<list name or id>

Update a list. A POST request can either do a complete update or modify rows in place using one of the following JSON examples.

Syntax

```
https://<username>:<password>@<host>/rest/decided_list/<list name or id>
```

POST

Update a list

The body of the request is a JSON object with the following fields.

Field	Required/Optional	Field type	Description
append_rows	Optional	JavaScript array	A two dimensional array. The new rows are appended to the existing rows. If there are values within the top level array that aren't arrays, they're transformed into an array to conform to the two dimensional requirement.
content	Optional	JavaScript array	A two dimensional array (array containing arrays containing simple values) which make the contents of your list. Replaces the current contents with the POSTed contents.
delete_rows	Optional	JavaScript array	An array of row indices (0 based). Each row is deleted.
name	Optional	string	The name of the list. The name must be unique.

Field	Required/Optional	Field type	Description
update_rows	Optional	JavaScript array	A JavaScript object where the keys are the row numbers to be updated and the values are an array that has the new content of the row. If you pass a single value instead of an array, it's transformed into an array to conform to the array requirement.

The following example does a complete replace.

Example request

```
{
    "content": [
        [
            "1.1.1.1"
        ],
        [
            "1.1.1.2"
        ],
        [
            "1.1.1.3"
        ],
        [
            "1.1.1.4"
        ],
        [
            "1.1.1.5"
        ],
        [
            "1.1.1.6"
        ],
        [
            "1.1.1.7"
        ],
        [
            "1.1.1.8"
        ]
    ],
    "name": "My IP address list"
}
```

The following example modifies parts of a list without replacing the entire thing. If you provide the `content` field, Splunk SOAR (On-premises) ignores other operations. If the individual cells of the table aren't strings, they're cast as strings.

Example request

```
{
    "append_rows": [
        [
            "2.2.2.13", "x", "y", "z"
        ]
    ],
    "delete_rows": [ 0, 11 ],
    "update_rows": {
        "0": ["1.1.1.x", "foo", "bar"],
        "5": ["1.1.1.y"],
        "11": ["test"]
    }
}
```

It's not possible to delete all rows from the list, and the delete_rows commands that attempt to do so results in an error. At least one row must be present in the list.

/rest/decided_list/<list name or id>/formatted_content<query parameters>

Return formatted data. You can get the contents of a list in a delimited format.

Syntax

```
https://<username>:<password>@<host>/rest/decided_list/<list name or id>/formatted_content<query
parameters>
```

GET

Get formatted data.

The Python CSV module can't handle unicode. Use JSON formatting if unicode must be supported.

The body of the request is a JSON object with the following fields.

Parameter	Required/Optional	Parameter type	Description
_output_format	Optional	"csv", "json", or "txt"	Output the contents either in JSON (default), comma delimited format, or text.
_fs	Optional	Single character	Value is used to separate fields in the same row. Comma by default.
_rs	Optional	Single character	Value is used to separate rows. Newline by default.

Example request

```
HTTP GET /rest/decided_list/<list name or ID>/formatted_content?_output_format=csv&_fs=,&_rs=%0A
```

Example response

A successful GET returns contents in comma delimited format.

```
A, B, C
D,,
E,,F
```

Notes endpoints

REST Note

Note endpoint to consolidate the note information that exists in multiple endpoints.

/rest/note

Create a note associated with a container.

Syntax

```
https://<username>:<password>@<host>/rest/note
```
POST

Create a general note associated with a container.

Request parameters
An argument string must include the following parameters:

Field	Required	Type	Description
container_id	required	integer	ID of the container to add the note to.
note_format	optional	string	Notes are displayed in either HTML or markdown. By default, `note_format` is set to `markdown` unless you pass in a different value. When you post an update of a note created before Splunk Phantom version 4.9, if you do not include the `note_format` as markdown, the note format defaults to HTML to maintain backwards compatibility with older notes.
title	required	string	The title of the note.
note_type	required	string	The type of the note: `general`.
content	required	string	The content of the note.
attachments	optional	list of integers	A list of the container attachment IDs to add to the note. A maximum of 10 attachments can be added to a note.

Example request
You can create a general note for container Id 93 by supplying a JSON formatted body.

```
curl -k -u admin:changeme https://localhost/rest/note \
-d '{"container_id": 93,
    "note_format": "markdown",
    "title": "something2",
    "note_type": "general",
    "content": "hello world",
    "attachments": [1, 2, 3]
}'
```
Example response
A successful request returns a success indicator and the ID of the newly created note.

```
{
   "success": true,
    "id": 4
}
```

POST

Create bulk container notes.

Request parameters

An argument string must include the following parameters:

Field	Required	Type	Description
container_id	required	integer	Id of the container to add the note to.
phase_id	optional	integer	If the container is a case, a phase_id can be provided to associate the note to a particular phase.
title	required	string	The title of the note.
note_type	required	string	The type of the note: `general`.
content	required	string	The content of the note.
note_format	optional	string	Notes are displayed in either HTML or markdown. By default, `note_format` is set to `markdown` unless you pass in a different value. When you post an update of a note created before Splunk Phantom version 4.9, if you do not include the `note_format` as markdown, the note format defaults to HTML to maintain backwards compatibility with older notes.

Example request

You can create many notes associated with many containers, such as multiple general notes for container Id 23, by supplying a JSON formatted body.

```
curl -k -u admin:changeme https://localhost/rest/note \
-d '[
    {
        "container_id": 23,
        "phase": 2,
        "title": "something2",
        "note_type": "general",
        "content": "hello world"
    },
    {
        "container_id": 23,
        "phase": 2,
        "title": "something2",
        "note_type": "general",
        "content": "hello world"
    }
]'
```

Example response

A successful request returns a success indicator and the IDs of the newly created notes.

```
[
    {
        "id": 17,
        "success": true
    },
    {
        "id": 18,
        "success": true
    }
]
```

POST

Create an artifact note.

Request parameters

An argument string must include the following parameters:

Field	Required	Type	Description
container_id	required	integer	The ID of the container.
phase_id	optional	integer	If the container is a case, a phase_id can be provided to associate the note to a particular phase.
author_id	optional	integer	ID of the user making the request.
title	required	string	The title of the note.
note_type	required	string	The type of the note: `artifact`.
artifact_id	required	string	A valid artifact ID.
content	required	string	The content of the note.

Example request

Create a note associated with an artifact by supplying a JSON formatted body with the `note_type` of `artifact`. Pass a valid artifact ID and phase ID. The container ID is set according to the artifact.

```
curl -k -u admin:changeme https://localhost/rest/note \
{
"container_id": 42270,
"phase_id": 1,
"author_id": 1,
"title": "my_note",
"note_type": "artifact",
"artifact_id": 155353,
"content": "hello world"
}
}'
```

Example response

A successful request returns a success indicator and the ID of the newly created note.

```
{"id": 7, "success": true}
```

POST

Create a task note.

Request parameters

An argument string must include the following parameters:

Field	Required	Type	Description
title	required	string	The title of the note.
note_type	required	string	The type of the note: `task`.
task_id	required	integer	A valid task Id.

Field	Required	Type	Description
content	required	string	The content of the note.
note_format	optional	string	Notes are displayed in either HTML or markdown. By default, `note_format` is set to `markdown` unless you pass in a different value. When you post an update of a note created before Splunk Phantom version 4.9, if you do not include the `note_format` as markdown, the note format defaults to HTML to maintain backwards compatibility with older notes.

Example request

Create a note associated with a task by supplying a JSON formatted body with the `note_type` of `task`. Pass a valid task Id. The phase is set according to the artifact.

```
curl -k -u admin:changeme https://localhost/rest/note \
-d '{
    "title": "my_note",
    "note_type": "task",
    "task_id":1,
    "content": "hello world"
}'
```

Example response

A successful request returns a success indicator and the IDs of the newly created note.

```
{
    "id": 24,
    "success": true
}
```

GET

Get a list of all notes for all items.

Example request

Get a list of notes, using paging parameters.

```
curl -k -u admin:changeme https://localhost/rest/note?page_size=5&page=0 -G -X GET
```

Example response

A successful GET will return a 200 response, and a JSON formatted list of notes.

```
{
        "count": 2,
        "data": [
            {
                "note_type": "general",
                "task": null,
                "container": 93,
                "title": "my note",
                "author": 1,
                "modified_time": "2019-03-26T23:50:00.451152Z",
                "content": "hello world",
                "create_time": "2019-03-26T23:50:00.384661Z",
                "artifact": null,
                "phase": 4,
                "id": 14
            },
            {
                "note_type": "general",
                "task": null,
                "container": 93,
                "title": "new note",
                "author": 1,
                "modified_time": "2019-03-27T00:16:08.220605Z",
                "content": "hello world",
                "create_time": "2019-03-27T00:14:09.090205Z",
                "artifact": null,
                "phase": 2,
                "id": 16
            }
        ],
        "num_pages": 1
    }
```

Example request

Return information about the note's attachments.

```
 /rest/note?_annotation_container_attachments
```

Example response

A successful return provides information about the size, name, and container attachment ID of the note.

```
{
'id': 144,
'title': 'Hello World',
'content': '<p>This is a Note </p>',
'create_time': '2020-04-01T18:08:58.667753Z',
'modified_time': '2020-04-01T18:08:58.673355Z',
'note_type': 'general',
'author': 332,
'phase': 373,
'container': 1148,
'task': None,
'artifact': None,
'container_attachments':
[{'container_attachment_id': 229, 'name': 'container_attachment_tGhDPDyIoOvA', 'size': 402},
 {'container_attachment_id': 230, 'name': 'container_attachment_wrQQiywwKqsj', 'size': 402}],
'task_name': None,
'artifact_name': None
}
```

/rest/note/<note_id>

List details of a single note.

Syntax

```
https://<username>:<password>@<host>/rest/note/<note_id>
```

GET

List details of a single note by ID.

Example request

List details of note Id 16.

```
curl -k -u admin:changeme https://localhost/rest/note/16 -G -X GET
```

Example response

A successful GET will return a 200 response, and a JSON formatted list of details.

```
{
        "note_type": "general",
        "task": null,
        "container": 93,
        "title": "new note",
        "author": 1,
        "modified_time": "2019-03-27T00:16:08.220605Z",
        "content": "hello world",
        "create_time": "2019-03-27T00:14:09.090205Z",
        "artifact": null,
        "phase": 2,
        "id": 16
    }
```

DELETE

Delete a note by ID.

Example request

Delete note Id 16.

```
curl -k -u admin:changeme https://localhost/rest/note/16 -X DELETE
```

Example response

A successful request will result in a 200 response returning the success message as JSON.

```
{
    "success": true
}
```

/rest/note optional filters

Filters for listing notes.

Syntax

```
https://<username>:<password>@<host>/rest/note?_filter_note_type
```

/rest/note?_filter_note_type=%22<artifact/task>%22

GET

List artifact and task notes.

Example request

List artifact notes. Notes created with /rest/artifact APIs also display in the /rest/note list views. Task notes have the associated task name in the 'artifact_name' field.

```
curl -k -u admin:changeme https://localhost/rest/note?_filter_note_type=%22artifact%22 -G -X GET
```

Example request
List task notes. Notes created with /rest/workflow_note APIs also display in the /rest/note list views. Task notes have the associated task name in the 'task_name' field

```
curl -k -u admin:changeme https://localhost/rest/note?_filter_note_type=%22task%22 -G -X GET
```

/rest/note optional query parameters

Search notes with the query parameter.

Syntax

```
https://<username>:<password>@<host>/rest/note?search=<search+term>
```

/rest/note?search=<search+term>

GET

Search notes with the query parameter.

You can search for the following:

- Note title
- Note content
- Task name for task notes
- Artifact name for artifact notes
- Artifact Id for artifact notes

Examples, using container_id=15: `/rest/note?_filter_container_id=15&search=note+title /rest/note?_filter_container _id=15&search=note+description /rest/note?_filter_container_id=15&search=artifact+title /rest/note?_filter _container_id=15&search=task+title /rest/note?_filter_container_id=15&search=43`

Example request
Search notes with the query parameter for "gorilla" and using container_id=15.

```
curl -k -u admin:changeme https://localhost/rest/note?_filter_container_id=15&search=gorilla -G -X GET
```

Example response
A successful request returns a success indicator and the items containing gorilla.

```
{
  "count": 1,
  "data": [
      {
          "artifact_name": null,
          "note_type": "task",
          "task": 4,
          "container": 15,
          "title": "Assigned to MgGilla.",
          "author": 1,
          "modified_time": "2019-05-16T17:09:00.437518Z",
          "content": "<p>This is a good case for the Gorilla!</p>",
          "create_time": "2019-05-16T17:09:00.428279Z",
          "artifact": null,
          "task_name": "Report incident response execution",
          "phase": 1,
          "id": 1
      }
  ],
  "num_pages": 1
}
```

Notification endpoints

REST Notification

Single notification endpoint that provides details about existing notifications in the system.

/rest/notification

List details of all notifications.

Syntax

```
https://<username>:<password>@<host>/rest/notification
```
GET

List details of notifications.

Example request
List details of notifications. The example shows the syntax for filtering and sorting.

```
curl -k -u uname:pwd https://localhost/rest/notification?_filter_is_read=False&pretty&sort=create
_time&order=desc&notifcation_type&due_time -G -X GET
```
Example response
A successful GET will return a 200 response, and a JSON formatted list of details. Since the example request shows the syntax for filtering and sorting, the example response shows filter responses. See REST Query Data for information about parameters like `_pretty` and `count`.

```
{
        "count": 6,
        "data": [

                {
                        "is_read": false,
                        "_pretty_to_user": "admin",
                        "_pretty_create_time": "0 minutes ago",
                        "ttl_minutes": 59,
                        "object_id": 62,
                        "to_user": 1,
                        "create_time": "2019-07-26T19:03:43.729000Z",
                        "_pretty_from_user": "",
                        "content_type": 10,
                        "from_user": null,
                        "message": "user initiated task",
                        "notification_type": "manualtask",
                        "due_time": " 2019-07-26T20:03:43.729000Z",
                        "id": 72
                },

                {
                        "is_read": false,
                        "_pretty_to_user": "admin",
                        "_pretty_create_time": "0 minutes ago",
        "ttl_minutes": 29,
        "object_id": 61,
        "to_user": 1,
        "create_time": "2019-07-26T19:03:17.058000Z",
        "_pretty_from_user": "",
        "content_type": 10,
        "from_user": null,
        "message": "prompt_1",
        "notification_type": "prompt",
        "due_time": " 2019-07-26T20:03:43.729000Z",
        "id": 71
},
{
        "is_read": false,
        "_pretty_to_user": "admin",
        "_pretty_create_time": "42 minutes ago",
        "ttl_minutes": 1439,
        "object_id": 60,
        "to_user": 1,
        "create_time": "2019-07-25T20:23:10.372000Z",
        "_pretty_from_user": "",
        "content_type": 10,
        "from_user": null,
        "message": "approval for block ip on asset 'zscaler'",
        "notification_type": "approval",
        "due_time": " 2019-07-26T20:03:43.729000Z",
        "id": 70
},
{
        "is_read": false,
        "_pretty_to_user": "admin",
        "_pretty_create_time": "43 minutes ago",
        "ttl_minutes": 1439,
        "object_id": 58,
        "to_user": 1,
        "create_time": "2019-07-25T20:22:55.118000Z",
        "_pretty_from_user": "",
        "content_type": 10,
        "from_user": null,
        "message": "approval for block ip",
        "notification_type": "actionreview",
        "due_time": " 2019-07-26T20:03:43.729000Z",
        "id": 68
},
{
        "is_read": false,
        "_pretty_to_user": "admin",
        "_pretty_create_time": "Today at 07:15 PM",
        "ttl_minutes": 43200,
        "object_id": 20,
        "to_user": 1,
        "create_time": "2019-07-25T19:15:56.065053Z",
        "_pretty_from_user": "adminTester",
        "content_type": 19,
        "from_user": 25,
        "message": "@admin hi",
        "notification_type": "containercomment",
        "id": 67
},
{
        "is_read": false,
        "_pretty_to_user": "admin",
                        "_pretty_create_time": "Jul 09 at 08:49 PM",
                        "ttl_minutes": 0,
                        "object_id": 697,
                        "to_user": 1,
                        "create_time": "2019-07-09T20:49:55.928000Z",
                        "_pretty_from_user": "",
                        "content_type": 73,
                        "from_user": null,
                        "message": "Task 'Determine functional impact' of phase 'Analysis and Containment'
for case 'HUD Container' has been assigned to you",
                        "notification_type": "workflowtask",
                        "due_time": " 2019-07-26T20:03:43.729000Z",
                        "id": 13
                }
        ],
        "num_pages": 1
}
```

The return values of note follow:

Field	Type	Description
notification_type	String	Always displays the notification type for all notifications, and distinguishes between the different types of approvals: `action reviews`, `approvals`, `manual tasks`, and `prompts`.
due_time	String	Displays notification due date for any notification that has a due date.
message	String	Displays message of notification no matter the type, because currently message is empty for approval notifications: `message`, and `approval for block ip`.

/rest/notification/<id>/detail_summary_view

List details of notifications for a particular container. The response varies depending on the notification type, which is a sub-type of notifications.

Syntax

```
https://<username>:<password>@<host>/rest/notification/<id>/detail_summary_view
```

GET

List details of notifications for a container where 21 is the container ID in the example request below, but not the ID in the example responses below.

Example request

List details of notifications.

```
curl -k -u uname:pwd https://localhost/rest/notification/21/detail_summary_view -G -X GET
```

Example response

A successful GET for approval notification type will return a 200 response, and a JSON formatted list of details.

```
{
        "update_time": "2019-08-19T21:43:58.892936Z",
        "container_id": 291,
        "time_left": 80128.535132,
        "next_owner": null,
        "action_name": "user initiated post ip action",
        "due_time": "2019-08-20T20:05:57.814000Z",
        "asset": {
                "action_whitelist": {},
                "validation": {},
                "tenants": [],
                "description": "Default Asset Configuration for AbuseIPDB",
                "tags": [],
                "type": "reputation",
                "primary_voting": 0,
                "product_version": "",
                "effective_user": 2,
                "product_name": "AbuseIPDB",
                "disabled": false,
                "token": null,
                "version": 1,
                "secondary_voting": 0,
                "configuration": {
                        "api_key": "<encrypted_key>"
                },
                "product_vendor": "AbuseIPDB",
                "id": 70,
                "name": "abuse_ip_db"
        },
        "action_type": "post ip",
        "container_name": "Possible Malicious Email",
        "owner": "admin",
        "notification_type": "approval",
        "type": "asset",
        "notification_targets": [{
                "app_id": 152,
                "parameters": [{
                        "comment": "Possibly malicious IP address",
                        "ip": "1.2.3.4",
                        "categories": "phishing"
                }],
                "assets": [
                        70
                ]
        }]
}
```

Example response

A successful GET for prompt notification type will return a 200 response, and a JSON formatted list of details.

```
{
        "playbook_repo": "local",
        "update_time": "2019-09-05T23:06:14.828588Z",
        "playbook_name": "Detect Malicious Domains",
        "container_id": 6355,
        "time_left": null,
        "next_owner": null,
        "action_name": "prompt_1",
        "container_name": "Malicious URL Request Attempt",
        "owner": "admin",
        "notification_type": "prompt",
        "escalated_approval": null,
        "due_time": "2019-09-05T23:06:14.709000Z",
        "jitc": {},
        "asset": null,
        "action_type": "prompt",
        "type": "manual",
        "notification_targets": [{
                "app_id": 0,
                "parameters": [{
                        "to": "admin@example.com",
                        "message": "Malicious domain detected",
                        "mins_to_act": 30,
                        "user_ids": [1],
                        "response_types": [{
                                "prompt": "General comments on domain",
                                "options": {
                                        "type": "message"
                                }
                        }, {
                                "prompt": "Have any users visited this malicious address?",
                                "options": {
                                        "type": "list",
                                        "choices": ["Yes", "No"]
                                }
                        }]
                }],
                "assets": []
        }]
}
```

Example response

A successful GET for manualtask notification type will return a 200 response, and a JSON formatted list of details.

```
{
        "update_time": "2019-09-05T01:34:58.002459Z",
        "container_id": 6325,
        "time_left": null,
        "next_owner": null,
        "action_name": "user initiated task",
        "container_name": "Possible Malicious Email",
        "owner": "admin",
        "notification_type": "manualtask",
        "escalated_approval": null,
        "due_time": "2019-09-05T01:34:57.833000Z",
        "jitc": {},
        "asset": null,
        "action_type": "task",
        "type": "manual",
        "notification_targets": []
}
```

Example response

A successful GET for actionreview notification type will return a 200 response, and a JSON formatted list of details.

```
{
        "playbook_repo": "local",
        "update_time": "2019-09-05T20:27:15.311964Z",
        "playbook_name": "Detect and Respond Against Malicious Domains",
        "container_id": 6354,
        "time_left": 32711.14501,
        "next_owner": null,
        "action_name": "block_ip",
        "container_name": "ASN Transaction",
        "owner": "admin",
        "notification_type": "actionreview",
        "escalated_approval": null,
        "due_time": "2019-09-06T08:27:15.301000Z",
        "jitc": {},
        "asset": null,
        "action_type": "block ip",
        "type": "parameter",
        "notification_targets": [{
                "app_id": 124,
                "parameters": [{
                        "is_source_address": "",
                        "ip": "1.1.1.1"
                }],
                "assets": [170]
        }]
}
```

The return values of note follow:

Field	Type	Description
asset	JSON Object	Can be empty depending on the notification type and if it contains an asset. See REST Assets for further information about assets.
container_id	String	The container Id of the playbook action run.
due_time	String	Time (UTC) when this action is due (time at which the SLA expires/expired).
next_owner	String	The next owner for an approval, such as admin.
notification_targets	JSON Object	JSON object containing a variety of parameters entered in response to `prompt`.
notification_type	String	`prompt`, `approval`, `manualtask`, `actionreview`.
owner	String	The current owner's display name, such as admin.
playbook_name	String	The playbook name.
playbook_repo	String	The name of the the playbook repository.
prompt	String	The options available to respond to a prompt such as: • `list`: a list of pre-selected strings • `message`: a user-inputted string • `range`: a number in the range specified by the prompt response It returns a dictionary that organizes the response answer percentage by response.
time_left	String	The due time minus the current time, in seconds.
type	String	Mapping for prettifying notification types, such as: • `prompt` -> `prompts` • `asset` -> `approvals` • `task` -> `manual tasks` • `parameter` -> `action reviews`

Playbook endpoints

REST Playbook

/rest/playbook/<id>

Updates a playbook.

Syntax

```
https://<username>:<password>@<host>/rest/playbook/<id>
```

POST

Toggle the status of a playbook between active and inactive. This determines whether or not the playbook will start automatically when an event is ingested with a label matching the playbook's label.

Request parameters
Playbooks are modified with the following parameters. No other fields can be updated.

Field	Required	Type	Description
active	Optional	boolean	Sets the playbook as active or inactive, when active, the playbook will run on ingested events with a corresponding label. Playbooks in draft mode cannot be marked active.
cancel_runs	Optional	boolean	Setting this to true when transitioning from active to inactive will cancel any current playbook runs associated with the playbook.

Example request
Set playbook Id 42 as inactive and cancel the playbook runs.

```
curl -k -u admin:changeme https://localhost/rest/playbook/42 \
-d '{
  "active": false,
  "cancel_runs": true
}'
```

Example response
A successful POST returns a descriptive message and success indicator.

```
{
    "message": "Operation successful",
    "success": true
}
```

/rest/import_playbook

POST

Imports a playbook.

Request parameters

Field	Required	Type	Description
playbook	Required	String	The base64 encoded playbook TAR file that you want to import.
scm/scm_id	Required	Name or ID of the repository	The repository where the playbook is saved.
force	Optional	boolean	Set to true to override an existing playbook in the same repository with the same name.

Example request

Imports a playbook.

```
curl -k -u admin:changeme https://localhost/rest/import_playbook \
-d '{
  "playbook": "H4sIAC0zs14C/+1c227bOBD1s76CUB+S7EaOJEu+BG2BohdgX4pFm31qA4GSKFuNTBoUncQN
/O9LUpJjObZE2dlkg4hA4wvPcIa30QwPa3QLp7MEebMELnxCrrqzReexi8lLv++IV2vgmuuvoliObXesXt91+j3LcayOafWtntUBZucJyjxlkA
LQmU0gZmS6E1dXn3dm9fpCiq7rmvinxdMZoQzk3ezSeYJSANPii6L+V0qwFlEyBSFkiMVTBPKa4vMpEH9DlDCohSgCBHtiiNlxQDCDMUb05FwD
vBSqQuTPx8dHK9wJCGCSoPDoRJM4iticYq1oLIpxnE7uWzsF6Xw6hXSxs9lcYq1dgXsDLiZxCqI5DlhMMODvs3oAI4Yo4O8BlFV8HCgCAREbha
Gwm4vnagGJJJZNUI4HEIdnJGshRNzKJJWgrLFcOoAY+KJVrjLgrYIJoqirldv2xHCDd6s+jRHz8qrjohdxBI4oSucJOwIxLklmA5LBImGQtMDL
0JvgH0Url/di9xoK0Tn24lAqKjVWFsnESoB0oxflyuNS8+9KlT82dF+egqzC4ysOvvsCk5SvuSiBjCGcfTzZYk15WZTVlxZa51UVtOn/xVJ4Uv
/vWK5tbfp/x+23/v8pyp1c+bqfkOAqWejnIJL7qfStdztNeI3+lr++f3sm/uo5IuBOf0yoENT
/wfmn+DcKVwACed3dakPqYtOWvslgCYKYf83oPFd+X0VC5ElLhBJ9ozZEaUDjmdjN26onMJ2I74f2MIR8pTkjFIyCcAT7/YHtO5Zt27DvDsKRFUQjO
/I35X+RGLMH5mZ2oSRJedWPB1Wi3G39VgqawqDvG5pKCEsgLqoQtkB8qEL0BOJbFcKp1SK85DUqL4rtQDxOBM6swjBG061DWYJ1fRhcjSmZ47A
WLAWiOJGr80228Sr6sxJJGSVXSAq5H
/uDQU+vlFme1pjMH+KEGj5sbHLvo+M4nw7UTubsbx6Ape+7Ig4z+Ec1KyiKhBHrA36qJGXciqnuuo9tdv7OJ+FCrQcCroSU6FjMji6G51RNgC1
mqBCplVgeNhosZnIH3anPQK+nOF1iNIcKWIZumfRNFx++XRy2KMc8zhLBslqPwjgV8YfQjQlGh6pGlBKqppnMYBAzodk8UGkciBxiCscNZ9FSm
cXbJMZX55N8x57FODjjD9azeDo+k09GT2jP0qduej0+dAAxYShtPHXZM1rbb5NUGJU17Ik4QKjZlVP+xGpJ5U+cIbNov+rxJySEc1ypfsOFYGg
QnCyANApcx+gG8OEC8JqnedDne1ilQWmTmPwacCrQCs9fPgxYZpEeI6vGH8ZS22REGoThFG2JnnbBC79YCedxNZl6RY9V4flAK0BFZKYIVeled
Ry5GVN6WcsKM3MPXsWwDUTWhq+BVD40dRLZ89CyrOEA9RwDQTgwnMC1jdHIjwwr9J3AR45ju5VrOsbyKS4C4csKmLDp0faRbIzIyYKJkmaVDcd
9LPKQjOEsuw5XNDioAKqsu8LVVoPmUx
/Rmvia0FBiqrpYxFw705Z15G5XXjXcs7z9mjBfBJwzxWeMXM2KQZ5allGOvZpFndsj+SiKFMPKlSyPFTCqd9Tbowcx0WZDobW0RyRKysLL06aD
yXcwSg4YzWbmaY/YCb256XzJp3H+4GjYYUjHaWMpKbmQC6CR3LLhcincV4Ii9lyzsdfQ7mF5vdUKFuuqqXfrqVpP1Xqqx
/dUNB5PXqiramD68tCzqL2ScOXu6fJUw6zZ8vmyMPezhaLrmMxTryBv6wLolYBKSJ5OyI23Crlrsug0/l1/4KNPkJzec+A6NcNyE4dMMBbD
/UaGpyXy0E3P8pMa5Lw28yiOJfMzwSpkfigQENj9LpR
/rjxP1m8gVUlSf4tlojUYhjri5XMt8fK1lnj51JImL500yY8tcspBMcN8bspk0+gnIUxi3JQv4RL/I7qkYDU+f/3Uchr
/LaehRlMgHL4WkqLirtouuuLBZbUMuP9ttUJ+z+tqhfj2+2oFmaJ2Y63AVl1ZyzAZrsmdtXu5ex11l9bKMpncM95a22ZO5bW1lsp6wE21VNbrp
bJG0PT9wTA03KHrGw7/a4xQ6Bi2HUE3suDQMXuKVJZWR0jsxY28LBLMdlVJsEq67HlYMFuRBWuprPaAuD0gfi0HxC2VpWhxS2W1nqr1VM
/oqVoqS+mc8AmorL7TUlnrydmTU1momkhqSGSJY+DnobHsR6KxGvFGy2c7Nsly8t4wHKLQDI2hb0aGg2DfGDkDxzDNXmQFvm2Go+EjXC9tU7o2
UGoDpTala1O6NqVrPVXrqdqUrk3pdqd0tq2S0vX3zFuapVGWOVTMo+yRs5dB6ynNX5j7Lr0mUelpahNzqVWYsMbpaFsg+mzBJjyTuEY0zbNeuz
tYs0xPgwmaip+KWPsPsvoa3DJHptk1u27XzTQsi5+coIhnjp74FSLZrGmbhukaZv/Cts7d4bnb7w7dgTlw/zTN89XdNz2kMGLeNGPn1n8FQ3qyMiuo
/5GpzFMNncFxlnpoy05b2tKWtryM8i/Q857rAFAAAA==",
        "scm": "local",
        "force": "true"
}'
```

Example response

A successful POST returns a success indicator and an import message.

```
{
    "success": true,
    "message": [
        true,
        "Playbook \"example_playbook\" imported"
    ]
}
```

/rest/playbook/<id>/export

GET

Exports a playbook.

Example request
Export a playbook.

```
curl -k -u admin:changeme --output <FILE> https://localhost/rest/playbook/1/export
```

Example response
A successful GET returns an x-gzip file to the location set in the `--output` flag.

/rest/playbook_resource_usage/<playbook_id>

GET

Gets the playbook run statistics for a specific playbook.

Request parameters

Field	Required	Type	Description
block_names	Optional	List	Filters on specified list of block names.
playbook_run_ids	Optional	List	Calculates the playbook run statistics for the specified playbook using only the specified list of playbook run IDs. `playbook_run_ids` and `time_range` are mutually exclusive.
time_range	Optional	List with 2 elements: `start_time` and `end_time`	Calculates the playbook run statistics for a given playbook only for playbook runs in that time range. Express the start and end of the `time_range` with only dates (['2022-09-13', '2022-09-14']) or with both dates and times (['2022-09-13T17:00', '2022-09-14T17:00']). `time_range` and `playbook_run_ids` are mutually exclusive.

The following requests return similar responses. A sample response is provided at the end of this list.

Example request using playbook ID
Get the playbook run statistics for playbook 27.

```
curl -k -u admin:changeme https://localhost/rest/playbook_resource_usage/27
```

Example request using time_range
Get the playbook run statistics for playbook 27 between 5:00pm on September 13, 2022 and 5:00pm on September 14, 2022.

```
curl -k -u admin:changeme https://localhost/rest/playbook_resource_usage/27?time_range=['2022-09-13T17:00',
'2022-09-14T17:00']
```

Example request using playbook ID and specific playbook run IDs
Get the playbook run statistics for playbook 27 for playbook runs 40 and 41.

```
curl -k -u admin:changeme https://localhost/rest/playbook_resource_usage/27?playbook_run_id=[40,41]
```

Example response using any of the previously mentioned example requests
Depending on the request used, the exact response might differ from this example response.
Each top-level name, like `geolocate_ip_1` and `on_finish`, corresponds to the name of a function in the playbook.

```
{
    "filter_1": {
        "avg_http_bytes_in_requests": 0,
        "avg_http_bytes_in_responses": 0,
        "avg_http_number_requests": 0,
        "avg_http_latency": "0:00:00",
        "avg_db_number_queries": 16,
        "avg_db_query_latency": "0:00:00.039000",
        "avg_duration": "0:00:00.025333",
        "times_called": 3,
        "times_succeeded": 3,
        "custom_function_scores": []
    },
    "geolocate_ip_1": {
        "avg_http_bytes_in_requests": 0,
        "avg_http_bytes_in_responses": 0,
        "avg_http_number_requests": 0,
        "avg_http_latency": "0:00:00",
        "avg_db_number_queries": 7,
        "avg_db_query_latency": "0:00:00.018000",
        "avg_duration": "0:00:00.022333",
        "times_called": 3,
        "times_succeeded": 3,
        "custom_function_scores": []
    },
    "join_random_get_requests": {
        "avg_http_bytes_in_requests": 0,
        "avg_http_bytes_in_responses": 0,
        "avg_http_number_requests": 0,
        "avg_http_latency": "0:00:00",
        "avg_db_number_queries": 5,
        "avg_db_query_latency": "0:00:00.008500",
        "avg_duration": "0:00:00.121833",
        "times_called": 6,
        "times_succeeded": 6,
        "custom_function_scores": []
    },
    "on_finish": {
        "avg_http_bytes_in_requests": 0,
        "avg_http_bytes_in_responses": 0,
        "avg_http_number_requests": 0,
        "avg_http_latency": "0:00:00",
        "avg_db_number_queries": 1,
        "avg_db_query_latency": "0:00:00.002000",
        "avg_duration": "0:00:00.002000",
        "times_called": 3,
        "times_succeeded": 3,
        "custom_function_scores": []
    },
    "on_start": {
        "avg_http_bytes_in_requests": 0,
        "avg_http_bytes_in_responses": 0,
        "avg_http_number_requests": 0,
        "avg_http_latency": "0:00:00",
        "avg_db_number_queries": 8,
        "avg_db_query_latency": "0:00:00.004000",
        "avg_duration": "0:00:00.058000",
        "times_called": 3,
        "times_succeeded": 3,
        "custom_function_scores": []
    },
    "random_get_requests": {
        "avg_http_bytes_in_requests": 0,
        "avg_http_bytes_in_responses": 14868,
        "avg_http_number_requests": 2,
        "avg_http_latency": "0:00:00.200000",
        "avg_db_number_queries": 3,
        "avg_db_query_latency": "0:00:00.007000",
        "avg_duration": "0:00:00.231000",
        "times_called": 3,
        "times_succeeded": 3,
        "custom_function_scores": []
    },
    "resource_score_cf_1": {
        "avg_http_bytes_in_requests": 0,
        "avg_http_bytes_in_responses": 0,
        "avg_http_number_requests": 0,
        "avg_http_latency": "0:00:00",
        "avg_db_number_queries": 3,
        "avg_db_query_latency": "0:00:00.006000",
        "avg_duration": "0:00:00.026000",
        "times_called": 3,
        "times_succeeded": 3,
        "custom_function_scores": [
            {
                "avg_http_bytes_in_requests": 0,
                "avg_http_bytes_in_responses": 0,
                "avg_http_number_requests": 0,
                "avg_http_latency": "0:00:00",
                "avg_db_number_queries": 0,
                "avg_db_query_latency": "0:00:00",
                "avg_duration": "0:00:00.000667",
                "times_called": 3,
                "times_succeeded": 3,
                "custom_function_id": 85,
                "custom_function_name": "resource_score_cf",
                "custom_function_scm": "local"
            }
        ]
    }
}
```

Example request using playbook ID and specific playbook block names
Get the playbook run statistics for all runs of playbook 27 between 5:00pm on September 13, 2022 and 5:00pm on September 14, 2022 for playbook block names `on_start` and `geolocate_ip_1`.

```
curl -k -u admin:changeme https://localhost/rest/playbook_resource_usage/27?time_range=['2022-09-13T17:00',
'2022-09-14T17:00']&block_names=['on_start', 'geolocate_ip_1']
```

You can also combine the block names with specific playbook run IDs or with a time_range.

- **Example request using playbook ID, specific playbook run IDs, and specific playbook block names**
  Get the playbook run statistics for playbook 27, run IDs 40 and 41, for playbook block names `on_start` and `geolocate_ip_1`.

  ```
  curl -k -u admin:changeme
  https://localhost/rest/playbook_resource_usage/27?playbook_run_id=[40,41]&block_names=['on_start',
  'geolocate_ip_1']
  ```
- **Example request using playbook ID, a specific time_frame, and specific playbook block names**
  Get the playbook run statistics for all runs of playbook 27 between 5:00pm on September 13, 2022 and 5:00pm on September 14, 2022 for playbook block names on_start and geolocate_ip_1.

  ```
  curl -k -u admin:changeme
  https://localhost/rest/playbook_resource_usage/27?time_range=['2022-09-13T17:00',
  '2022-09-14T17:00']&block_names=['on_start', 'geolocate_ip_1']
  ```

Example response using time_range and specific playbook block names
A successful GET returns all of the playbook run statistics for the specified playbook within the specified date/time span, for the specific block names.

```
{
    "on_start": {
        "avg_http_bytes_in_requests": 0,
        "avg_http_bytes_in_responses": 0,
        "avg_http_number_requests": 0,
        "avg_http_latency": "0:00:00",
        "avg_db_number_queries": 8,
        "avg_db_query_latency": "0:00:00.004000",
        "avg_duration": "0:00:00.058000",
        "times_called": 3,
        "times_succeeded": 3,
        "custom_function_scores": []
    },
     "geolocate_ip_1": {
        "avg_http_bytes_in_requests": 0,
        "avg_http_bytes_in_responses": 0,
        "avg_http_number_requests": 0,
        "avg_http_latency": "0:00:00",
        "avg_db_number_queries": 7,
        "avg_db_query_latency": "0:00:00.018000",
        "avg_duration": "0:00:00.022333",
        "times_called": 3,
        "times_succeeded": 3,
        "custom_function_scores": []
    }
}
```
Notes:

- Custom functions have section titles including `_cf` and are associated with the blocks that call them. Statistics can show custom code that includes multiple, consecutive `phantom.custom_function()` calls.
- Statistics are not provided for action/app runs, but are provided for the playbook block that runs the action/apps. Statistics show the time taken for the block to complete, not for the action to complete. Even if the action fails, the statistics might show the block as a success because the block successfully *started* the action.

Role management endpoints

REST Roles and Permissions

With the exception of "container_labels","repository", and "tenant" you can only have one permission of a given name/type per role. You can have multiple of these types of permissions if they specify access to different names (specified in the "extra" field).

Users that have no named object permissions in any role they are assigned to will have access to all of them. As soon as a named object permission is added to any role, then users will be restricted to only the whitelisted objects.

For example, if a user only has roles with the "basic permissions" (no roles containing any container_labels, repository, or tenant permissions) then that user will have access to ALL of the objects for each of those types. The moment the user is given a named object permission (for example, a "container label" of "archer incident") then the access is ONLY to archer incident containers, and none of the others. Though, if only container_labels permissions are specified in one or more of the user's roles, then the user would still have access to ALL repositories and tenants.

Also note that roles are additive, so if a user previously had access to all container labels because none of the roles specified any, and then you grant the user a role that specifies "archer incident", now the user only has archer incident. If you add two roles, one that has "archer incident" and one that has "campaign", then that user now has those two and no others.

/rest/role

Manage roles and permissions.

Syntax

```
https://<username>:<password>@<host>/rest/role
```

POST

Create a new role with permissions.

The body of the request is a JSON object with the following fields.

Field	Required	Type	Description
add_users	optional	JSON Array	An array of integers that are the user IDs for users to be added to this role. (Used to update a record.)
description	required	string	Description of the new role.
immutable	read-only	boolean	True if the role may not be modified. This is true for standard roles shipped with Splunk SOAR (On-premises) and false for user-created roles.
name	required	string	Name of the new role.
permissions	optional	JSON Array	An array of JSON Objects describing the permissions for the role. This value overrides update_permissions and replaces existing permissions with the provided set. See Permissions below. Note that this is optional but a role with no permissions does not provide any

Field	Required	Type	Description
			functionality.
remove_users	optional	JSON Array	Array of integers that are user IDs to be removed from the role. (Used to update a record.)
update_permissions	optional	JSON Array	Used to add or modify existing set of permissions. Only one permission can exist with a given name per role. If update_permissions contains an existing name, the matching permission will be updated. Otherwise a new permission will be created with the new name. See Permissions below.
users	optional	JSON Array	Array of integers that are user IDs. This sets or completely replaces the user's assigned to this role. Overrides add_users and remove_users.

Permissions objects are defined by the following JSON object.

Field	Required	Type	Description
edit	optional	boolean	Permission allows modification of the specified part of the system.
execute	optional	boolean	Permission allows execution of actions or playbooks. (Only applies to the "playbooks" permission.)
extra	optional	boolean	Field for specifying additional information about the permission. For the "container_labels", "repository", and "tenant" permissions, the value of the field should be the object name. For example, for the local Splunk SOAR (On-premises) repository, use "local" as the value for the extra field.
object_id	optional	integer	Used for the "tenant" and "repository" permissions, in which case it is the id corresponding to the name of the tenant or repository provided in extra field. When used to set repository permissions in multi-tenant systems, the object_id is required.
delete	optional	boolean	Permission allows deletion of the specified part of the system.
name	required	string	Name of the permission. This specifies which part of the system the permissions are applied to. Valid values are as follows: • "apps": Allows view/modification/install of apps. • "assets": Allows view/modification of assets. • "containers": Allows view/modification of containers and artifacts. Can also see action and playbook executions. • "container_labels": Restricts interaction with containers to whitelisted labels. Makes use of the "extra" field. Can be duplicated. • "repository": Restricts users with this role to whitelisted repositories. Makes use of the "extra" field. Can be duplicated. • "tenant": Restricts users with this role to whitelisted tenants. Makes use of the "extra" field. Can be duplicated. • "playbooks": Allows view/modification/execution of playbooks and execution of actions. • "system_settings": Allows view/modification of system settings. • "users_roles": Allows view/modification of users and roles.
view	optional	boolean	Permission allows user to view of the specified part of the system.

Example request

Create a new role with permissions and assign it to users.

```
curl -k -u admin:changeme https://localhost/rest/role \
-d '{
    "name": "newrole",
    "description": "Description of newrole",
    "users": [4, 7, 13, 169],
    "permissions": [
        {
            "name": "containers",
            "view": true,
            "edit": false,
            "delete": false,
            "extra": "incidents"
        },
        {
            "name": "playbooks",
            "view": true,
            "edit": true,
            "execute": true
        }
    ]
}'
```

Example response

A successful POST will return a success indicator and the Id of the newly created role.

```
{"id": 7, "success": true}
```

/rest/role/<role_id>

Manage an existing role or permission.

POST

Update an existing role.

The body of the request is a JSON object with the same fields as /rest/role

Example request

Update role Id 7 with revised users and permissions.

```
curl -k -u admin:changeme https://localhost/rest/role/7 \
-d '{
    "add_users": [20, 38],
    "remove_users": [13, 169],
    "update_permissions": [
        {
            "name": "containers",
            "edit": true
        },
        {
            "name": "playbooks",
            "view": false,
            "edit": false,
            "execute": false
        }
    ]
}'
```

Example response

A successful POST will return a success indicator and the Id of the revised role.

```
{"id": 7, "success": true}
```

GET

Retrieve data for one role.

Example request
Retrieve role data for role Id 1.

```
curl -k -u admin:changeme https://localhost/rest/role/1 -G -X GET
```

Example response
A successful GET will return back a JSON formatted list of key names and data.

This example role has delete, edit, and view for "Label Permissions" of "archer incident"; "Repository Permissions" for "local"; and "Tenant Permissions" for "DefaultTenant".

With no additional roles, this user would only be able to see the one specific named object for each of the three types.

```
{
    execute: "deny",
    name: "container_label",
    extra: "archer incident",
    edit: "allow",
    object_id: null,
    role: 7,
    content_type: null,
    delete: "allow",
    id: 53,
    view: "allow"
},
{
    execute: "deny",
    name: "repository",
    extra: "local",
    edit: "allow",
    object_id: 2,
    role: 7,
    content_type: 51,
    delete: "allow",
    id: 54,
    view: "allow"
},
{
    execute: "deny",
    name: "tenant",
    extra: "DefaultTenant",
    edit: "allow",
    object_id: 0,
    role: 7,
    content_type: 20,
    delete: "allow",
    id: 55,
    view: "allow"
}
```

Run action endpoints

REST Run Action

Run an action using a REST request.

/rest/action_run

Run an action against a container.

Syntax

```
https://<username>:<password>@<host>/rest/action_run
```
POST

Run an action against a container.

Request parameters
Actions are modified with the following parameters.

Field	Required	Type	Description
action	required	String.	Name of the action to be run. (e.g. "file reputation", "block ip" etc.)
container_id	required	integer	Id of the container to run the action on.
name	required	String.	Name for this action.
targets	required	JSON Array of Objects.	Specifies one or more combinations of assets/app/parameters. See below for details.
type	optional	String.	Categorization for the action.
assets	Optional depending on command. (See the documentation for the App/action being run.)	JSON array of strings.	The assets to run the command on. Can be asset names or asset IDs.
app_id	required	String.	Identifies the App that will run the command. Can be the app GUID or the app's numeric ID.
parameters	Optional depending on command (See the documentation for the App/action being run.)	JSON Array of Objects	An array of sets of parameters. Each entry in the list is a JSON Object containing the parameters for action being run.

Actions will not always require both assets and parameters, although at least one will be required. If the assets or the parameters value is not required, that value can be an empty array or it can be omitted. To see what is required for an action, see the App documentation for that action.

Example request
Run an action named "file reputation" against container Id 42 .

```
curl -k -u admin:changeme https://localhost/rest/action_run \
-d '{
        "action": "file reputation",
        "container_id": 42,
        "name": "my action",
        "targets": [{
                "assets": [
                        "my_reversinglabs_asset"
                ],
                "parameters": [{
                        "hash": "98dbaeb6d46bd09eca002e1f2b6f3e76fd3222cd"
                }],
                "app_id": 12
        }],
        "type": "investigative"
}'
```

Example response

A successful POST will return back a descriptive message, a success indicator and the ID of the newly created action run.

```
{
    "action_run_id": 2,
    "message": "New action queued.",
    "success": true
}
```

Example request

Actions can be run on multiple assets using a specified App. The App will take a specific set of parameters for that action. The targets section of the run action JSON will allow you to run several variations of the same action in one call. For example, to run the same command on 3 different hashes:

```
curl -k -u admin:changeme https://localhost/rest/action_run \
-d '{
        "action": "file reputation",
        "container_id": 42,
        "name": "my action",
        "targets": [{
                "assets": [
                        "my_reversinglabs_asset"
                ],
                "parameters": [{
                                "hash": "<hash 1>"
                        },
                        {
                                "hash": "<hash 2>"
                        },
                        {
                                "hash": "<hash 3>"
                        }
                ],
                "app_id": 12
        }],
        "type": "investigative"
}'
```

/rest/action_run/<id>

Once you have run an action, you may want to check its status or cancel it.

Syntax

```
https://<username>:<password>@<host>/rest/action_run/<id>
```
GET

Check the status of an existing run action.

Request string
An argument string must include the `action_run_id`.

Response
A success or failure message.

Example request
Check the status of action run Id 2.

```
curl -k -u admin:changeme https://localhost/rest/action_run/2 -G -X GET
```
Example response
A successful GET will return a JSON object with the action details.

```
{
        "action": "whois domain",
        "assign_time": "2016-01-15T22:15:58.062000Z",
        "cancelled": "",
        "cb_fn": "whois_domain_cb",
        "close_time": "2016-01-15T22:16:00.798179Z",
        "comment": "",
        "container": 24,
        "create_time": "2016-01-15T22:15:58.062000Z",
        "creator": null,
        "due_time": "2016-01-12T14:32:30.810000Z",
        "exec_delay_secs": 0,
        "exec_order": 0,
        "handle": "",
        "id": 47,
        "message": "1 actions succeeded",
        "name": "automated action 'whois domain' of 'whois_app' playbook",
        "owner": null,
        "playbook": 42,
        "playbook_run": 46,
        "status": "success",
        "targets": [{
                "app_id": 23,
                "assets": [
                        119
                ],
                "parameters": [{
                        "domain": "amazon.com"
                }]
        }],
        "type": "investigate",
        "update_time": "2016-01-15T22:16:00.798179Z",
        "version": 1
}
```
The return values follow:

Field	Type	Description

action	String	Action that was run.
assign_time	ISO 8601 formatted timestamp	For manual actions, this is the time (UTC) at which the action was assigned to the owner.
cancelled	String	If the playbook was cancelled, this field will contain a message indicating why.
cb_fn	String	For Internal use.
close_time	ISO 8601 formatted timestamp	Timestamp (UTC) when this action completed/ended.
comment	String	For manual actions. Initial comment describing the task.
container	Integer	ID of the container on which this action was run.
create_time	ISO 8601 formatted timestamp	Timestamp (UTC) when this action was created.
creator	Integer	ID of the user who created the action or null if created from automation.
due_time	ISO 8601 formatted timestamp	Timestamp (UTC) when this action is due (time at which the SLA expires/expired).
exec_delay_secs	Integer	If action is to be carried out in the future, this is the initial delay before running the action.
exec_order	Integer	For Internal use.
handle	Integer	For Internal use.
id	Integer	ID of this action run.
message	String.	Indicates progress or results of action.
name	String	Name assigned to this action run.
owner	Integer	ID of the user who ran the playbook or null if there is no owner.
playbook	Integer	ID of the playbook used for this run.
playbook_run	Integer	If run from a playbook, ID of the playbook_run.
start_time	ISO 8601 formatted timestamp	Timestamp (UTC) of when the playbook run was started.
status	String	Status of the action run. Status equals one of the following values: • failed • pending • running • success
targets	JSON	See targets section above.
type	String	The type of action. Types include... • contain • correct • investigate • manual • etc.
update_time	ISO 8601 formatted timestamp	Timestamp (UTC) of when the action run was last active.
version	Integer	For internal use. Schema version.

POST

Cancel a running action.

Example request

Cancel action run Id 2.

```
curl -k -u admin:changeme https://localhost/rest/action_run/2 \
-d '{
    "cancel": true
}'
```

Example response

A successful POST will return a descriptive message and success indicator.

```
Success response body:
{
    "cancelled": true,
    "message": "<detail>"
}

Error response body:
{
    "failed": true,
    "message": "<reason>"
}
```

/rest/action_run/<id>/app_runs

Retrieve the app action results.

Syntax

```
https://<username>:<password>@<host>/rest/action_run/<id>/app_runs
```

GET

Retrieve the app action results.

Example request

```
curl -k -u admin:changeme https://localhost/rest/action_run/2/app_runs
```

Example response

A successful GET returns the action app results if available.

```
 {
    "count": 1,
    "num_pages": 1,
    "data": [
        {
            "id": 15,
            "action": "whois domain",
            "action_run": 2,
            "asset": 5,
            "app": 190,
            "app_name": "WHOIS",
            "app_version": "2.0.7",
            "container": 337,
            "end_time": "2020-07-07T21:35:55.645000Z",
            "exception_occured": false,
            "message": "'user initiated whois domain action' on asset 'whois': 1 action succeeded. (1)For
```

```
Parameter: {\"domain\":\"example.com\"} Message: \"Domain: example.com, Organization: Example, Inc., Name: 
Host, Example Inc., City: San Jose, Country: US\"",
            "playbook_run": null,
            "start_time": "2020-07-07T21:35:53.231000Z",
            "status": "success",
            "version": 1
        }
    ]
}
```

The return values follow:

Field	Type	Description
id	Integer	ID of this action run.
action	String	Action that was run.
action_run	Integer	The number of actions in the action run.
asset	JSON array of strings.	The assets to run the command on. Can be asset names or asset IDs.
app	Integer	The numeric ID of the app the action was run on.
app_name	String	Name of the app.
app_version	Integer	Version of the app.
container	Integer	Numeric ID of the container.
end_time	ISO 8601 formatted timestamp	The time the action run completed.
exception_occured	Boolean	Whether an exception occurred or not.
message	String	Indicates progress or results of action.
playbook_run	Integer	If run from a playbook, ID of the playbook_run.
start_time	ISO 8601 formatted timestamp	Timestamp (UTC) of when the playbook run was started.
status	String	Request the playbook to be run. One of the following: • failed • pending • running • success
version	Integer	For internal use. Schema version.

/rest/app_run

Get the results of an app's run, based on parameters you specify.

Syntax

```
https://<username>:<password>@<host>/rest/app_run
```

GET

Check the app action run result data.

Example request

Check the status of the app action run results.

```
/rest/app_run/{app_run.id}/action_result
```
Add pagination query parameters.

```
/rest/app_run/1/action_result?page=0&page_size=5
```
Example response

A successful GET will return a JSON object with the app action run results.

```
{
   "count":1,
   "num_pages":1,
   "data":[
      {
         "data":[
            {
               "latitude":20,
               "longitude":77,
               "time_zone":"Asia/Kolkata",
               "country_name":"India",
               "continent_name":"Asia",
               "country_iso_code":"IN"
            }
         ],
         "status":"success",
         "message":"Country: India",
         "summary":{
            "country":"India"
         },
         "parameter":{
            "ip":"103.230.84.239",
            "context":{
               "guid":"22070b58-353f-4ba7-94ed-b4a8228fe93f",
               "artifact_id":0,
               "parent_action_run":[

               ]
            }
         }
      }
   ]
}
```
The return values follow:

Field	Type	Description
context	Object	The context around how the app action is executed. For example, the server and artifact the app action is invoked against and the action run using this app.
count	Integer	The number of app runs.
data	Object	Contains the data from the app run.
ip	String	The IP address of the app action run.
message	String	Indicates progress or results of the action.
parameter	Object	Parameters that are passed into an app action to be executed.
num_pages	Integer	Pagination of the app run.

Field	Type	Description
status	Boolean	Whether or not the app action run was successful or not. Options include either "success" or "failure".
summary	Object	The summary of the action run result.

/rest/app_run/{app_run.id}

Check the results of a single app run.

GET

Check the results of a single app run.

Example request
Check the results of a single app run.

```
/rest/app_run/{app_run.id}
```

Example response
A successful GET will return a JSON object with the results of a single app run.

```
{
   "id":7,
   "action":"check python version",
   "action_run":6,
   "asset":229,
   "app":191,
   "app_name":"QA App Py3",
   "app_version":"2.0.3",
   "container":331,
   "end_time":"2022-01-26T23:49:40.871000Z",
   "exception_occured":false,
   "extra_data":[

   ],
   "message":"'check python version' on asset 'qa_app_py3': 1 action succeeded. (1)For Parameter:
{\"context\":{\"artifact_id\":0,\"guid\":\"99d948fc-6490-4143-ae2b-f93682fc6e11\",\"parent_action_run\":[]}}None
Message: \"Running in python version 3.6.15+, which matches the expected python version 3\"",
   "result_summary":{
      "total_objects":1,
      "total_objects_successful":1
   },
   "result_data":[
      {
         "data":[

         ],
         "status":"success",
         "message":"Running in python version 3.6.15+, which matches the expected python version 3",
         "summary":{

         },
         "parameter":{
            "context":{
               "guid":"99d948fc-6490-4143-ae2b-f93682fc6e11",
               "artifact_id":0,
               "parent_action_run":[
                ]
            }
        }
    }
   ],
   "playbook_run":null,
   "start_time":"2022-01-26T23:49:40.307000Z",
   "status":"success",
   "version":1,
   "effective_user":1,
   "node_guid":"ec05c5c5-0e22-4f3e-a48d-48a67f498dc3",
   "automation_broker":null
}
```
The return values follow:

Field	Type	Description
action	String	The description of the action being run on the app.
action_run	Integer	The ID of the related action run.
app	Integer	The ID of the related app.
app_name	String	The name of the related app.
app_version	Integer	The version of the related app.
asset	Integer	The ID of the related asset.
automation_broker	ID	The ID of the automation broker, if applicable.
container	Integer	The ID of the container that the app ran on.
context	Object	The context around how the app action is executed. For example, the server and artifact the app action is invoked against and the action run using this app.
effective_user	Integer	The user ID of the user executing the action.
end_time	ISO 8601 formatted timestamp	Timestamp (UTC) when the app run ended.
exception_occurred	Boolean	Whether or not an exception occurred to block the app run. Available options are "true" or "false".
id	Integer	The ID of the app run.
message	String	Indicates progress or results of the app run.
node_guid	String	The unique identifier of the server that is running the action.
parameter	Object	Parameters that are passed into an app action to be executed.
playbook_run	Integer	If run from a playbook, the ID of the playbook run.
result_summary	Object	The summary of the result of the app run.
start_time	ISO 8601 formatted timestamp	Timestamp (UTC) of when the action run was started.
status	Boolean	Indicates the status of the action run. Available options are "success" or "failure".
version	Integer	The version of the app.

Run playbook endpoints

REST Run Playbook

Run a playbook against a container or check the playbook status.

/rest/playbook_run

Run a playbook against a container.

Syntax

```
https://<username>:<password>@<host>/rest/playbook_run
```
POST

Run a playbook against a container.

Request parameters
You can use the REST API to run a playbook against a container.

Field	Required	Type	Description
container_id	required	integer	Id of the container to run the playbook on.
playbook_id	required	integer or string	The name of the playbook in the form "/". Also can be the Id of the playbook to run.
scope	optional	list of integers or string	Can be a string containing "all" or "new" or a list of integers containing artifact Ids. This allows the playbook to be run against a specific set of artifacts.
inputs	optional	JSON	If specified, this value must be a JSON object with one or more key/value pairs, passed as an input for the child playbook that is called, specified by the "playbook_id" field. Be sure both of the following are true: • The child playbook that is called must be an input playbook, not an Automation playbook. • The inputs you provide must be valid for the child playbook.
run	required	Boolean true.	Request the playbook to be run.

Example request
This is a request to run the playbook "file_analysis" against the container with id 42. You can infer that the "file_analysis" playbook is an input playbook by the presence of the "inputs" key in the request.

```
curl -k -u admin:changeme https://localhost/rest/playbook_run \
-d '{
        "container_id": 42,
        "playbook_id": "local/file_analysis",
        "scope": "new",
        "run": true,
        "inputs": {
            "file_hash": "09ca7e4eaa6e8ae9c7d261167129184883644d07dfba7cbfbc4c8a2e08360d5b",
            "file_name": "hello.txt"
    }
}'
```
Example response

A successful POST will return back an acknowledgement indicator and the ID of the newly created playbook run. Note that this API will only queue the playbook to run. The result indicates that the request is received, however the playbook may not successfully run (if there is a problem with the playbook, for example).

```
{
    "playbook_run_id": "1",
    "received": true
}
```
/rest/playbook_run/<id>

Check the status of a playbook run.

Syntax

```
https://<username>:<password>@<host>/rest/playbook_run/<id>
```
GET

Check the status of a playbook run.

Example request

Get the status of playbook run Id 46.

```
curl -k -u admin:changeme https://localhost/rest/playbook_run/46 -G -X GET
```
Example response

A successful request returns a 200 response and a JSON object containing the status for the playbook.

```
{
        "action_exec": [],
        "cancelled": "",
        "container": 24,
        "id": 46,
        "last_artifact": 160,
        "log_level": 1,
        "message": "{\"message\":\"\",\"playbook_run_id\":46,\"result\":[{\"action\":\"whois
ip\",\"app_runs\":[{\"action\":\"whois ip\",\"action_run_id\":46,\"app_message\":\"Registry: arin\\nASN:
8075\\nCountry: US\\nNets:\\nRange: 134.170.0.0 - 134.170.255.255\\nAddress: One Microsoft
Way\",\"app_name\":\"WHOIS\",\"app_run_id\":55,\"asset_name\":\"whois\",\"parameter\":\"{\\\"ip\\\":
\\\"134.170.188.221\\\"}\",\"per_parameter_status\":\"success\"}],\"close_time\":\"2016-01
-15T22:15:58.05319+00:00\",\"create_time\":\"2016-01-15T22:15:55.252+00:00\",\"id\":46,\"message\":\"1
actions succeeded\",\"status\":\"success\",\"type\":\"investigate\"},{\"action\":\"whois
domain\",\"app_runs\":[{\"action\":\"whois domain\",\"action_run_id\":47,\"app_message\":\"Organization:
Amazon Technologies, Inc.\\nDomain: amazon.com\\nCity: Reno\\nName: Hostmaster, Amazon Legal
Dept.\\nCountry:
US\",\"app_name\":\"WHOIS\",\"app_run_id\":56,\"asset_name\":\"whois\",\"parameter\":\"{\\\"domain\\\":
\\\"amazon.com\\\"}\",\"per_parameter_status\":\"success\"}],\"close_time\":\"2016-01
-15T22:16:00.798179+00:00\",\"create_time\":\"2016-01-15T22:15:58.062+00:00\",\"id\":47,\"message\":\"1
actions succeeded\",\"status\":\"success\",\"type\":\"investigate\"}],\"status\":\"success\"}\n",
        "owner": 4,
        "playbook": 42,
        "start_time": "2016-01-15T22:15:55.171000Z",
        "status": "success",
        "test_mode": false,
        "update_time": "2016-01-15T22:16:00.813027Z",
        "version": 1
}
```
The return values follow:

Field	Type	Description
action_exec	JSON	For internal use.
cancelled	String	If the playbook was cancelled, this field will contain a message indicating why.
container	Integer	ID of the container used in this run.
id	Integer	ID of this playbook run.
last_artifact	Integer	For internal use.
log_level	Integer	Indicates if logging was enabled for this playbook run.
message	String	Message. The string value contains a JSON object with details about the results of the action run. NOTE: Contents and/or data type of this field is likely to change in the future.
owner	Integer	ID of the user who ran the playbook or null if there is no owner.
playbook	Integer	ID of the playbook used for this run.
start_time	ISO 8601 formatted timestamp	Timestamp (UTC) of when the playbook run was started.
status	String	Request the playbook to be run. One of the following: • failed • running • success
test_mode	Boolean	Request the playbook to be run.
update_time	ISO 8601 formatted timestamp	Timestamp (UTC) of when the playbook run was last active.
version	Integer	For internal use. Schema version.

POST

Cancel a running playbook.

Example request

Cancel playbook run Id 46.

```
curl -k -u admin:changeme https://localhost/rest/playbook_run/46 \
-d '{
    "cancel": true
}'
```

Example response

A successful POST will return a descriptive message and success indicator.

```
Success response body:
{
    "cancelled": true,
    "message": "<detail>"
}

Error response body:
{
    "failed": true,
    "message": "<reason>"
}
```

Severity endpoints

REST Severity

You can manage the severities using REST. Supported methods are GET, POST, and DELETE.

Splunk SOAR (On-premises) ships with three predefined severity names: **High**, **Medium**, and **Low**. Your organization might need additional levels of severity to match your business processes. Additional severity names can be defined by a Splunk SOAR (On-premises) administrator.

Rules for severity names:

- You can have up to 10 active severity names.
- A severity name can consist only of the ASCII characters A-Z, a-z, numerals 0-9, dash (-), or underscore (_).
- Severity names can be up to 20 characters long.
- Severity names cannot be edited. To change a severity name, delete it and recreate the severity name.
- The default severity names **High**, **Medium**, and **Low** can be deleted. However, even if they are deleted, your ingestion apps will still be able to use the severity names **High**, **Medium**, and **Low**.
- A severity name which has been deleted can be reactivated by creating a new severity with the same name.

Severity names are stored in Splunk SOAR (On-premises)'s internal database. Deleting a severity name from the active severity list does not remove that severity name from the database.

Deleting a severity name does not change the severity of a case, event, or artifact. Changing a severity name does not update closed events, cases, or artifacts.

Deleted severity names show in search results, the Analyst Queue, Investigation, and dashboard widgets where appropriate. Severity names which have been deleted are shown in all these areas using strikethrough text.

Deleted severities have a few other impacts, such as:

- You cannot filter by disabled severities in Analyst Queue.
- Using the graphical user interface, you cannot create a container with a deleted severity or change a container's severity to a deleted severity.
- Events and artifacts with deleted severities will appear lower than those with active severities in sorted lists in the Analyst Queue and Investigation.

To maintain backwards compatibility with apps and existing playbooks, if the severity names High, Medium, or Low have been deleted, ingestion apps and the REST API can still assign the severity High, Medium, and Low to events, containers, or artifacts.

/rest/severity

Get a list of all severities.

Syntax

```
https://<username>:<password>@<host>/rest/severity
```

Usage details
You must have the "View System Settings" permission to list severities. You You need "System Settings Edit" permissions to create or modify a severity.

GET

Get a list of all severities.

Example request
Get a list of available CEF.

```
curl -k -u admin:changeme https://localhost/rest/severity -G -X GET
```

Example response
A successful request will result in a 200 response and a JSON formatted list of severities.

```
{
    "count": 3,
    "data": [
        {
            "disabled": false,
            "name": "high",
            "color": "red",
            "modified_time": "2019-05-29T20:22:05.809886Z",
            "id": 1,
            "is_default": false,
            "create_time": "2019-05-29T20:22:05.809371Z",
            "order": 0
        },
        {
            "disabled": false,
            "name": "medium",
            "color": "yellow",
            "modified_time": "2019-05-29T20:22:05.811406Z",
            "id": 2,
            "is_default": true,
            "create_time": "2019-05-29T20:22:05.811153Z",
            "order": 1
        },
        {
            "disabled": false,
            "name": "low",
            "color": "green",
            "modified_time": "2019-05-29T20:22:05.812411Z",
            "id": 3,
            "is_default": false,
            "create_time": "2019-05-29T20:22:05.812242Z",
            "order": 2
        }
    ],
    "num_pages": 1
}
```

POST

Create a severity.

Request string
An argument string must include the following parameters:

Field	Required	Type	Description
color	optional	string	One of the following color choices: • red • orange • yellow • green • light_blue • blue • purple • light_grey • dark_grey • pink
name	required	string	Name of the severity.
is_default	required	bool	A status with "is_default"' set to True becomes the default severity.

Example request

You can add aseverity by supplying a JSON formatted body.

```
curl -k -u admin:changeme https://localhost/rest/severity \
-d '{
    "color" : "red",
    "name" : "urgent"
}'
```

Example response

A successful request will result in a 200 response, returning the severity's Id and success as JSON.

```
{
    "id": 6,
    "success": true
}
```

/rest/severity/<severity id>

Modify an existing severity.

Syntax

```
https://<username>:<password>@<host>/rest/severity/<severity id>
```

Usage details

You need "System Settings Edit" permissions to create or modify a severity.

POST

Modify an existing a severity, such as changing the urgent severity color from "red" to "orange".

Request string

An argument string must include the following parameters:

Field	Required	Type	Description
color	optional	string	One of the following color choices:

Field	Required	Type	Description
			• red • orange • yellow • green • light_blue • blue • purple • light_grey • dark_grey • pink
name	required	string	Name of the severity.
is_default	required	bool	A status with "is_default'" set to True becomes the default severity.

Example request

You can add aseverity by supplying a JSON formatted body.

```
curl -k -u admin:changeme https://localhost/rest/severity \
-d '{
    "color" : "orange",
    "name" : "urgent"
}'
```

Example response

A successful request will result in a 200 response, returning the severity's Id and success as JSON.

```
{
    "id": 6,
    "success": true
}
```

Source control repository endpoints

REST Source Control Repository

Sync a source control repository such as https://github.com/phantomcyber/playbooks.git or file:////opt/phantom/scm/git/local.

/rest/scm/<scm_id>

Sync an existing source control repository.

Syntax

```
https://<username>:<password>@<host>/rest/scm/<scm_id>
```
POST

Sync an existing source control repository by ID.

Example request
Sync source control repository Id 3. The body of the request is a JSON Object with a "pull" command. The "force" key can be used to cause the pull operation to do a more aggressive sync, discarding local changes or conflicts.

```
curl -k -u admin:changeme https://localhost/rest/scm/3 \
-d '{
  "pull": true,
  "force": true
}'
```
Example response
A successful POST will return an array of changes that were made. Each entry has two parts. Part 1 is the type of change, either (A)dded, (D)eleted or (M)odified. Part 2 is the name of the playbook that was changed.

```
[
    ["M", "playbook1"],
    ["D", "playbook2"],
    ["M", "playbook3"],
    ["A", "playbook4"]
]
```

Status endpoints

REST Status

You can manage status using REST. Supported methods are GET, POST, and DELETE.

Statuses are grouped into three categories or types: New, Open, and Resolved. Your business processes may require additional statuses, so Splunk SOAR (On-premises) lets you to create additional statuses in each category, up a to maximum 10 total statuses.

Rules for status labels:

- There must be at least one active status label for each status type, **New**, **Open**, and **Resolved**.
- You can have a maximum of 10 status labels.
- The labels **New**, **Open**, and **Closed** can be deleted, removing them from the active list. These labels cannot be renamed because they are required for backward compatibility with apps and playbooks.
- The name of a status label can consist only of the ASCII characters A-Z, a-z, numerals 0-9, dash (-), or underscore (_).
- A status label cannot be more than 20 characters long.

A status label's JSON object includes an "id" field populated with an integer. The integer can be used in many REST API queries in place of the status label's name, such as in filtering. See Query for Data.

To maintain backwards compatibility with apps and existing playbooks, if the status labels New, Open, or Closed have been deleted, ingestion apps and the REST API can still assign the statuses New, Open, and Closed to containers.

/rest/container_status

List all status labels.

Syntax

```
https://<username>:<password>@<host>/rest/container_status
```
GET

Get a list of all status labels.

Example request
Get a list of all status labels.

```
curl -k -u admin:changeme https://localhost/rest/container_status -G -X GET
```
Example response
A successful GET will return a 200 response, and a JSON formatted list of statuses.

```
{
    "count": 4,
    "data": [
        {
            "disabled": false,
            "name": "closed",
            "is_mutable": false,
            "modified_time": "2019-05-29T20:22:09.466124Z",
            "id": 3,
            "is_default": true,
            "create_time": "2019-05-29T20:22:09.465988Z",
            "order": 0,
            "status_type": "resolved"
        },
        {
            "disabled": false,
            "name": "new",
            "is_mutable": false,
            "modified_time": "2019-05-29T20:22:09.464784Z",
            "id": 1,
            "is_default": true,
            "create_time": "2019-05-29T20:22:09.464504Z",
            "order": 0,
            "status_type": "new"
        },
        {
            "disabled": false,
            "name": "ripe",
            "is_mutable": true,
            "modified_time": "2019-05-30T00:06:49.758771Z",
            "id": 6,
            "is_default": false,
            "create_time": "2019-05-30T00:06:49.756331Z",
            "order": 1,
            "status_type": "new"
        },
        {
            "disabled": false,
            "name": "open",
            "is_mutable": false,
            "modified_time": "2019-05-29T20:22:09.465676Z",
            "id": 2,
            "is_default": true,
            "create_time": "2019-05-29T20:22:09.465519Z",
            "order": 0,
            "status_type": "open"
        }
    ],
    "num_pages": 1
}
```
POST

Create a new status label.

Request parameters The "System Settings Edit" permission is required to add statuses. An argument string must include the following parameters:

Field	Required	Type	Description
name	required	string	The name of the new status.
status_type	required	string	One of "New", "Open", or "Resolved".
is_default	required	bool	A status with "is_default'" set to True becomes the default status.

Field	Required	Type	Description

Example request

You can create a new status label by supplying a JSON formatted body.

```
curl -k -u admin:changeme https://localhost/rest/container_status \
-d '{
    name: "fresh",
    status_type: "new"
}'
```

Example response

A successful request will result in a 200 response returning the new status id and success as JSON.

```
{
    "id": 4,
    "success": true
}
```

DELETE

Delete a status label.

Example request

The "System Settings Edit" permission is required to delete statuses. Delete a status label with Id 4.

```
curl -k -u admin:changeme https://localhost/rest/container_status/4 -X DELETE
```

Example response

A successful request will result in a 200 response returning the new status id and success as JSON.

```
{
    "id": 4,
    "success": true
}
```

System settings endpoints

REST System Settings

Update a subset of system settings such as the authentication providers.

/rest/system_settings

Update authentication providers.

Syntax

```
https://<username>:<password>@<host>/rest/system_settings
```
POST

Update the authentication provider settings by replacing the entire configuration or by replacing parts of it.

Request parameters
Authentication providers are modified with the following parameters.

Field	Required	Type	Description
auth_settings	optional	Javascript Object	A complex data structure containing all authentication providers. The entirety of "auth_settings" must be submitted in a single post. Partial updates are not supported.

Each section is modified with the following keys at the top level.

Field	Required	Type	Description
enabled	required	Boolean	Flag indicating if this section is enabled (applies to all of each ldap/saml2/openid sections).
providers	required	Javascript array of objects	An array of the provider configurations. JSON for each provider described below.
type	required	String	Type should match the name of the section. One of saml2, openid or ldap.

LDAP providers are modified with the following keys.

Field	Required	Type	Description
name	required	String	The name of the provider configuration entry.
type	required	String	Type of provider, should match the section. One of saml2, openid or ldap.
id	required	String	ID of the provider, this should be a GUID-like entry.
host	required	String	IP or hostname of the provider.
domain	required	String	LDAP domain.
enabled	required	String	Flag indicating if the provider is enabled.
username	required	String	Username for the service account used to query the provider.

Field	Required	Type	Description
force_ssl	optional	Boolean	If set to true, will only connect using ldaps. Otherwise will use ldap.
password	optional	String	The password for the service account used to query the provider. Required if not using a credential manager to store your password.
use_credential_manager	optional	Boolean	Flag indicating if the service account password should be retrieved from the credential manager.
cm_key	optional	String	Key for identifying the password in the credential manager. Required if using a credential manager.
cm_path	optional	String	Path for identifying the password in the credential manager. Required if using a credential manager.
cm_safe	optional	String	Identifies the Safe that contains the credentails in Cyberark. Required if using Cyberark as a credential manager.
test_username	optional	String	Username for testing LDAP access and queries.
test_group	optional	String	Used to verify the test_username is in the expected group. Requires the test_username to be set.
user_attr_map	optional	JSON Array of JSON Objects	Each entry of the array should contain two key-value pairs. The "django_attr" key should have one of the following values. • username • email • first_name • last_name • title • location • time_zone The "external_attr" key should contain the name of the LDAP attribute used to populate the django attribute.
group_role_mappings	optional	JSON Array of JSON Objects	Each entry of the array should contain two key-value pairs. The "role" key should contain the numeric ID of the Splunk SOAR (On-premises) role. The "group" key should contain the name of the LDAP group that translates to the Splunk SOAR (On-premises) group

SAML2 providers are modified with the following keys.

Field	Required	Type	Description
name	required	String	The name of the provider configuration entry.
type	required	String	Type of provider, should match the section. One of saml2, openid or ldap.
id	required	String	ID of the provider, this should be a GUID-like entry.
issuer_id	required	String	The issuer ID (URI) given by your provider
metadata_url	required	String	The URL to retrieve SAML metadata. This is the preferred method of obtaining provider metadata since it should always be up-to-date.
metadata_xml	required	String	The XML containing the SAML provider metadata. This is used for out-of-band configuration.
single_sign_on_url	required	String	URL used to gain user consent/authorization from the identity provider. Users will be redirected here for login when using SAML2.
enabled	required	String	Flag indicating if the provider is enabled.

Field	Required	Type	Description
phantom_base_url	optional	String	Base URL for the Splunk SOAR (On-premises) instance. Not required only if the FQDN has been set in Company Settings.
force_ssl	optional	Boolean	Flag indicating if SSL is required (defaults to FALSE).
group_key	optional	String	Used to locate group data in the SAML assertion, treated as a mapping object, e.g. {"groups": ["group1", "group2", "group3"]} (defaults to "groups").
user_attr_map	optional	JSON Array of JSON Objects	Each entry of the array should contain two key-value pairs. The "django_attr" key should have one of the following values. Can be used if attributes are not auto-populated correctly. • username • email • first_name • last_name • title • location • time_zone The "external_attr" key should contain the name of the LDAP attribute used to populate the django attribute.
group_role_mappings	optional	JSON Array of JSON Objects	Each entry of the array should contain two key-value pairs. The "role" key should contain the numeric ID of the Splunk SOAR (On-premises) role. The "group" key should contain the name of the LDAP group that translates to the Splunk SOAR (On-premises) group.

OpenID providers are modified with the following keys.

Field	Required	Type	Description
name	required	String	The name of the provider configuration entry.
type	required	String	Type of provider, should match the section. One of saml2, openid or ldap.
id	required	String	ID of the provider, this should be a GUID-like entry.
client_id	required	String	Client ID given by the identity provider.
enabled	required	String	Flag indicating if the provider is enabled.
service_discovery_url	required	String	URL where the issuer provides service information.
phantom_base_url	optional	String	Base URL for the Splunk SOAR (On-premises) instance. Not required only if the FQDN has been set in Company Settings.
force_ssl	optional	Boolean	Username for the service account used to query the provider.
client_secret	optional	String	The client_secret. Required if not using a credential provider and token_endpoint_auth_method is set to client_secret_post.
use_credential_manager	optional	Boolean	Flag indicating if the service account password should be retrieved from the credential manager.
cm_key	optional	String	Key for identifying the password in the credential manager. Required if using a credential manager (Cyberark/Hashicorp/Thycotic).
cm_path	optional	String	Path for identifying the password in the credential manager. Required if using a credential manager (Cyberark/Hashicorp/Thycotic).
cm_safe	optional	String	Identifies the Safe that contains the credentials in Cyberark. Required only if using Cyberark as a credential manager.

Field	Required	Type	Description
cm_mapping	optional	String	The fieldName in the Thycotic secret which has the credential. Required only if using Thycotic as a credential manager.
scope	optional	String	Comma delimited list of scopes.
resource	optional	String	Used to identify the protected resource which contains user claims, information about the user. Leave blank to use the Userinfo endpoint
user_attr_map	optional	JSON Array of JSON Objects	Each entry of the array should contain two key-value pairs. The "django_attr" key should have one of the follwoing values. • username • email • first_name • last_name • title • location • time_zone `The "external_attr" key should contain the name of the LDAP attribute used to populate the django attribute.`
group_role_mappings	optional	JSON Array of JSON Objects	Each entry of the array should contain two key-value pairs. The "role" key should contain the numeric ID of the Splunk SOAR (On-premises) role. The "group" key should contain the name of the LDAP group that translates to the Splunk SOAR (On-premises) group.
group_key	optional	String	Used to locate group data in the data returned by the OpenID identity provider, treated as a mapping object, e.g. {"groups": ["group1", "group2", "group3"]} (defaults to "groups").

Example request
Completely replace the authentication settings by doing the following:

```
curl -k -u admin:changeme https://localhost/rest/system_settings \

        "auth_settings": {
                "ldap": {
                        "enabled": true,
                        "providers": [{
                                "cm_key": "value",
                                "cm_path": "secret/ldap",
                                "domain": "example.com",
                                "enabled": true,
                                "group_role_mappings": [],
                                "host": "1.1.1.1",
                                "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
                                "name": "LDAP Provider 1",
                                "password": "<password>",
                                "test_group": "HR",
                                "test_username": "alice",
                                "type": "ldap",
                                "use_credential_manager": true,
                                "user_attr_map": [],
                                "username": "ldap-service-account"
                        }],
                        "type": "ldap"
                },
                "openid": {
                        "enabled": true,
                        "providers": [{
                                        "client_id": "<client id>",
                                        "client_secret": "<client secret>",
                                        "enabled": true,
                                        "force_ssl": false,
                                        "group_key": "role",
                                        "group_role_mappings": [{
                                                "group": "HR",
                                                "role": "31"
                                        }],
                                        "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
                                        "name": "OpenID - ADFS",
                                        "phantom_base_url": "http://example.com",
                                        "resource": "companyresource",
                                        "scope": "openid",
                                        "service_discovery_url": "https://example.com",
                                        "token_endpoint_auth_method": "private_key_jwt",
                                        "type": "openid",
                                        "user_attr_map": [{
                                                "django_attr": "username",
                                                "external_attr": "given_name"
                                        }]
                                },
                                {
                                        "client_id": "<client id>",
                                        "client_secret": "<client secret>",
                                        "enabled": true,
                                        "group_role_mappings": [{
                                                "group": "special group",
                                                "role": "31"
                                        }],
                                        "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
                                        "name": "OpenID - Okta",
                                        "phantom_base_url": "http://example.com",
                                        "service_discovery_url": "https://example.com",
                                        "type": "openid",
                                        "user_attr_map": []
                                }
                        ],
                        "type": "openid"
                },
                "saml2": {
                        "enabled": true,
                        "providers": [{
                                "enabled": true,
                                "group_key": "role",
                                "group_role_mappings": [{
                                        "group": "HR",
                                        "role": "31"
                                }],
                                "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
                                "issuer_id": "http://example.com/provider",
                                "metadata_url":
"https://example.com/federationmetadata/2007-06/federationmetadata.xml ",
                                "metadata_xml": "",
                                "name": "SAML2 Provider 1",
                                "phantom_base_url": "https://example.com",
                                "single_sign_on_url": "https://example.com",
                                "type": "saml2",
                                "user_attr_map": []
                        }],
                        "type": "saml2"
                }
        }
}'
```

Example response

A successful POST returns a success or failure indicator and a message for each command.

```
    },
    "message": "At least one update operation succeeded. See individual responses.",
    "success": true,
    "auth_settings": {
        "failed": true,
        "message": "ID for ldap provider \"LDAP Provider 1\" does not appear to be a GUID"
    }
}
```

Example request

Modify parts of the auth settings list without replacing the entire thing. If "contents" is given, other operations are ignored. If the individual "cells" of the table are not strings, they are coerced to strings.

```
curl -k -u admin:changeme https://localhost/rest/system_settings \
-d '{
        "ldap": {
                "enabled": true,
                "providers": [{
                        "cm_key": "value",
                        "cm_path": "secret/ldap",
                        "domain": "example.com",
                        "enabled": true,
                        "group_role_mappings": [],
                        "host": "1.1.1.1",
                        "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
                        "name": "LDAP Provider 1",
                        "password": "<password>",
                        "test_group": "HR",
                        "test_username": "alice",
                        "type": "ldap",
                        "use_credential_manager": true,
                        "user_attr_map": [],
                        "username": "ldap-service-account"
                }],
                "type": "ldap"
        },
        "openid": {
                "enabled": true,
                "providers": [{
                                "client_id": "<client id>",
                                "client_secret": "<client secret>",
                                "enabled": true,
                                "force_ssl": false,
                                "group_key": "role",
                                "group_role_mappings": [{
                                        "group": "HR",
                                        "role": "31"
                                }],
                                "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
                                "name": "OpenID - ADFS",
                                "phantom_base_url": "http://example.com",
                                "resource": "companyresource",
                                "scope": "openid",
                                "service_discovery_url": "https://example.com",
                                "token_endpoint_auth_method": "private_key_jwt",
                                "type": "openid",
                                "user_attr_map": [{
                                        "django_attr": "username",
                                        "external_attr": "given_name"
                                }]
                        },
                        {
                                "client_id": "<client id>",
                                "client_secret": "<client secret>",
                                "enabled": true,
                                "group_role_mappings": [{
                                        "group": "special group",
                                        "role": "31"
                                }],
                                "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
                                "name": "OpenID - Okta",
                                "phantom_base_url": "http://example.com",
                                "service_discovery_url": "https://example.com",
                                "type": "openid",
                                "user_attr_map": []
                        }
                ],
                "type": "openid"
        },
        "saml2": {
                "enabled": true,
                "providers": [{
                        "enabled": true,
                        "group_key": "role",
                        "group_role_mappings": [{
                                "group": "HR",
                                "role": "31"
                        }],
                        "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
                        "issuer_id": "http://example.com/provider",
                        "metadata_url":
"https://example.com/federationmetadata/2007-06/federationmetadata.xml ",
                        "metadata_xml": "",
                        "name": "SAML2 Provider 1",
                        "phantom_base_url": "https://example.com",
                        "single_sign_on_url": "https://example.com",
                        "type": "saml2",
                        "user_attr_map": []
                }],
                "type": "saml2"
        }
}'
```

Example response

A successful POST returns a success or failure indicator and a message for each command.

```
  {
    "message": "At least one update operation succeeded. See individual responses.",
    "success": true,
    "auth_settings": {
        "failed": true,
        "message": "ID for ldap provider \"LDAP Provider 1\" does not appear to be a GUID"
    }
}
```

/rest/system_settings/features

Enable or disable system settings features.

Syntax

```
https://<username>:<password>@<host>/rest/system_settings/features
```

Usage details

The system setting edit permissions are required to change this setting.

POST

Enable system settings features.

Request parameters

Features are modified with the following parameters.

Field	Required	Type	Description
feature	required	String	The feature: clickable_url, multi_tennancy, indicators.
set_feature_enabled	required	Boolean	True or False.

Example request

Enable indicators as follows:

```
curl -k -u admin:changeme https://localhost/rest/system_settings/features \
-d '{
        "feature": "indicators",
        "set_feature_enabled": "True"
}'
```

Example response

A successful POST returns a success or failure indicator and a message for each command.

```
{
        "message": "Updated feature enabled state",
        "enabled": "True",
        "feature": "indicators",
        "success": true
}
```

/rest/system_settings optional query parameters

/rest/system_settings?sections

Get the status of feature system settings sections.

Syntax

```
https://<username>:<password>@<host>/rest/system_settings?sections
```

GET

Get the status of system settings.

Example request
Get the status of the mobile system setting.

```
curl -k -u admin:changeme https://localhost/rest/system_settings?sections=["mobile"] -G -X GET
```

Example response
A successful GET will return back a JSON formatted list of key names and their data.

```
{
  "mobile": {
    "enabled": false
  }
}
```

User management endpoints

REST User

Two types of users may be provisioned in Splunk SOAR (On-premises), "automation" users and "normal" users. Automation users are excluded from general user queries by default. To include automation users you can either use a filter `/rest/ph_user?_filter_type__in=["normal", "automation"]` or as a convenience, use the GET parameter include_automation, for example: `/rest/ph_user?include_automation`.

/rest/ph_user

Manage users.

Syntax

```
https://<username>:<password>@<host>/rest/ph_user
```

POST

Create users.

The body of the request is a JSON object with the following fields.

Field	Required	Type	Description
add_roles	optional	JSON Array	An array of integers that are role IDs or role names. Must be only role ids or only exact role names. New roles to be added to the existing set of roles for the user. (Used to update a record.)
allowed_ips	optional	JSON Array	A whitelist of IPs (CIDR netmask allowed) for automation/service accounts to restrict Splunk SOAR (On-premises) server access with the associated token. Default is an empty list, or no access.
default_label	optional	string	The default label to apply to containers created by an automation user. Only applies to automation/service accounts.
default_tenant_id	optional	integer	The default tenant to apply to containers created by an automation user. Only applies to automation/service accounts. Is overridden by the asset's tenant id or an explicit tenant id.
email	optional	string	Use this field if the username is not an email address or the user should be contacted at an email address that is different than the username. If this field does not contain a valid email address, the user will not receive email notifications.
first_name	optional	string	User's given name.
type	optional	string	The type of authentication to confirm user identity, "normal" (used for local users), "ldap", "openid", or "saml2".
last_name	optional	string	User's family name.
location	optional	string	Physical location of th user. Usually an office or city location.
password	optional	string	Password for the user in clear text (communication happens over HTTPS for security). This field is required when creating a non-AD user. Otherwise is optional. Cannot be used on an AD user.
remove_roles	optional	JSON Array	An array of integers that are role IDs or role names. Must be only role ids or only exact role names. List of roles to be removed from the existing set of roles for the user. (Used to update a record.)
roles	optional		

Field	Required	Type	Description
		JSON Array	Array of integers that are role IDs or role names. Must be only role ids or only exact role names. This sets or completely replaces the user's roles. Overrides add_roles and remove_roles.
time_zone	optional	string	Time zone user works in. See https://en.wikipedia.org/wiki/List_of_tz_database_time_zones for a list of time zones.
title	optional	string	User's title.
type	optional	string	Set to "automation" when creating a automation/service account. Can only be set when creating a user, ignored on update.
username	required	string	User's login name.
2fa	optional	string	String indicating two factor authentication for the user. Currently the only valid values for this are "duo" and null.
2fa_username	optional	string	Username to use for 2FA if not using the user's username.

Example request
Create a standard user.

```
curl -k -u admin:changeme https://localhost/rest/ph_user \
-d '{
    "username": "john.doe@example.com",
    "first_name": "John",
    "last_name": "Doe",
    "email": "john.doe@example.com",
    "title": "Automation Engineer",
    "location": "Palo Alto",
    "time_zone": "US/Pacific",
    "password": "cleartextpassword",
    "roles": [8, 14, 23],
    "2fa": "duo",
    "2fa_username": "johndoe"
}'
```

Example request
Create an automation/service user. Fewer fields are required for this type of user.

```
curl -k -u admin:changeme https://localhost/rest/ph_user \
-d '{
    "allowed_ips": ["10.10.0.0/16"],
    "username": "service_account",
    "roles": ["Automation"],
    "type": "automation"
}'
```

Example request
Create an LDAP user. Fewer fields are required for this type of user.

```
curl -k -u admin:changeme https://localhost/rest/ph_user \
-d '{

    "username": "john.doe@example.com",
    "email": "john.doe@example.com",
    "time_zone": "US/Pacific",
    "type": "ldap",
    "roles": [8, 14, 23],
    "2fa": "duo",
    "2fa_username": "johndoe"
}'
```

Example response

A successful response includes the numeric Id given to the user.

```
{
    "id": 34,
    "success": true
}
```

GET

Retrieve user data.

Example request

Retrieve user data. If no user Id is provided a list of users will be returned.

```
curl -k -u admin:changeme https://localhost/rest/ph_user -G -X GET
```

Example response

A successful GET will return back a JSON formatted list of key names and data.

```
{
    "count": 2,
    "data": [
        {
            "username": "george.tailor@splunk.com",
            "first_name": "George",
            "last_name": "Tailor",
            "roles": [],
            "title": null,
            "prevent_login": false,
            "is_active": true,
            "time_zone": null,
            "email": "george.tailor@splunk.com",
            "is_superuser": false,
            "is_staff": false,
            "last_login": null,
            "location": null,
            "default_label": null,
            "default_tenant": null,
            "type": "normal",
            "id": 8,
            "date_joined": "2017-07-29T00:44:39.951851Z"
        },
        {
            "username": "automation",
            "first_name": "",
            "last_name": "",
            "allowed_ips": ["127.0.0.1"],
            "roles": ["Automation"],
            "title": null,
            "prevent_login": false,
            "is_active": true,
            "time_zone": "UTC",
            "email": "",
            "is_superuser": false,
            "is_staff": false,
            "last_login": "2017-08-07T21:14:25.910672Z",
            "location": null,
            "default_label": "events",
            "default_tenant": 0,
            "type": "automation",
            "id": 4,
            "date_joined": "2017-07-29T00:44:40.131816Z"
          },
    ],
    "num_pages": 1
}
```

/rest/ph_user/<userid>

Manage a user.

Syntax

```
https://<username>:<password>@<host>/rest/ph_user/<userid>
```

GET

Retrieve user data for one user.

Example request
Retrieve user data for user Id 1.

```
curl -k -u admin:changeme https://localhost/rest/ph_user/1 -G -X GET
```

Example response
A successful GET will return back a JSON formatted list of key names and data.

```
{
        "last_name": "",
        "is_staff": true,
        "external_update_needed": false,
        "default_label": null,
        "id": 1,
        "date_joined": "2019-11-13T22:50:37.397186Z",
        "first_name": "",
        "title": null,
        "default_tenant": null,
        "is_superuser": true,
        "last_login": "2019-11-18T21:13:41.220935Z",
        "location": null,
        "type": "normal",
        "email": "root@localhost",
        "username": "admin",
        "prevent_login": false,
        "is_active": true,
        "onboarding_state": {
                "redirect_onboarding": false
        },
        "externally_managed": false,
        "roles": ["Administrator"],
        "time_zone": "UTC",
        "show_onboarding": false
}
```

POST

Update an existing user.

The body of the request is a JSON object with the same fields as /rest/ph_user plus the following optional fields for convenience.

Field	Required	Type	Description
add_roles	optional	array	The Id of the role to add, based on /rest/role.
remove_roles	optional	array	The Id of the role to remove, based on /rest/role.

Example request
Add and remove roles for user Id 1.

```
curl -k -u admin:changeme https://localhost/rest/ph_user/1 \
-d '{
    "add_roles": [6],
    "remove_roles": [8, 14]
}'
```

User settings endpoints

REST User Settings

Combination of settings containing PhUser and Profile attributes.

/rest/user_settings

Manage a special serialized user object that contains PhUser and Profile attributes.

Syntax

```
https://<username>:<password>@<host>/rest/user_settings
```

Usage
If you're logged in as admin or as a regular user when you send a GET request, the server retrieves the active session user and handles the request based on that. This means that you can only view or update your own settings. Only non-automation users can POST to the API.

GET

Get a list of user settings.

Example request
Get a list of admin user settings.

```
curl -k -u admin:changeme https://localhost/rest/user_settings -G -X GET
```

Example response
A successful GET will return a JSON formatted list of user settings.

```
{
        "username": "admin",
        "last_name": "",
        "notify_my_reassigned": true,
        "notify_my_expiredsla": true,
        "show_onboarding": false,
        "is_admin": true,
        "password_constraints": "There are no password constraints configured.",
        "local_docs": false,
        "notify_any_closed": false,
        "notify_any_nearsla": false,
        "notify_any_reassigned": false,
        "first_name": "",
        "redirect_onboarding": false,
        "title": null,
        "notify_my_assigned": true,
        "time_zone": "UTC",
        "notify_my_mentions": true,
        "theme": "dark",
        "notify_my_nearsla": true,
        "location": null,
        "display_relative_time": true,
        "type": "normal",
        "email": "root@localhost",
        "notify_any_breachedsla": false
}
```

The parameters are described in the POST.

POST

Update your own user settings.

Request parameters

An argument string must include the following parameters in the body:

Field	Required	Type	Description
email	optional	string	The user email.
first_name	optional	string	The user's given name.
last_name	optional	string	The user's family name.
title	optional	string	The user's title.
local_docs	optional	bool	Sets the the user's default choice for viewing documentation: • false = online - links to docs.splunk.com • true = offline - points to a local copy of the docs in PDF format
location	optional	string	Physical location of the user. Usually an office or city location.
notify_my_assigned	optional	bool	Indicates if the user has the notifications check box checked for My Events: "Event or task assigned to me."
notify_my_nearsla	optional	bool	Indicates if the user has the notifications check box checked for My Events: "Event SLA expiring soon."
notify_my_expiredsla	optional	bool	Indicates if the user has the notifications check box checked for My Events: "Event SLA expired."
notify_my_reassigned	optional	bool	Indicates if the user has the notifications check box checked for My Events: "Event reassigned."
notify_my_mentions	optional	bool	Indicates if the user has the notifications check box checked for My Events: "Collaboration notifications."
notify_any_nearsla	optional	bool	Indicates if the user has the notifications check box checked for All Events: "Event SLA expiring soon."
notify_any_breachedsla	optional	bool	Indicates if the user has the notifications check box checked for All Events: "Event SLA expired."
notify_any_reassigned	optional	bool	Indicates if the user has the notifications check box checked for All Events: "Event reassigned."
notify_any_closed	optional	bool	Indicates if the user has the notifications check box checked for All Events: "Event resolved."
password_constraints	optional	string	Indicates if there are password constraints configured.
time_zone	optional	string	Time zone where the user works. See https://en.wikipedia.org/wiki/List_of_tz_database_time_zones for a list of time zones.
theme	optional	string	light or dark
display_relative_time	optional	bool	Indicates if the user has the notifications check box checked for "Display Relative Timestamps."

Example request

Set a new email address and theme by supplying a JSON formatted body.

```
curl -k -u admin:changeme https://localhost/rest/user_settings \
-d '{
        "email": "username@splunk.com",
        "theme": "dark",
        "local_docs": true
}'
```

Example response

A successful request will result in a success message indicating that the settings are updated.

```
{
        "message": "Settings updated",
        "success": true
}
```

Search endpoints

REST Search

To do a search, an HTTP GET is made to `/rest/search`.

Syntax

```
https://<username>:<password>@<host>/rest/search
```
GET

Search query

Request parameters
The endpoint accepts the following parameters.

Query	Required	The query to be searched
categories	optional	Comma separated list of categories to search. Valid categories follow: • container • artifact • asset • app • action • playbook • docs If categories are not provided, the query will be searched across all of them.
tenant	optional	Id of the tenant for whom you want to filter search results.
page	optional	Positive integer. Returned results are paginated. This parameter requests a specific page.
page_size	optional	Positive integer. Returned results are paginated. This parameter determines how many results returned per-page. Use "0" for all results.

Example request
Search query for 'maxmind' in the 'app' and 'docs' categories.

```
curl -k -u admin:changeme https://localhost/rest/search?query=maxmind&categories=app,docs -G -X GET
```
Example response
A successful GET will return back a JSON formatted list of key names and data.

```
{
    count: 10,
    num_pages: 1,
    results: [
        {
            url: "https://10.1.16.99/docs/app_reference/maxmind_c566e153-3118-4033-abda-14dd9748c91a",
            category: "app",
            verbose: "App",
            match: {
                message: ""
            }
        },
        {
            url: "https://10.1.16.99/docs/rest/search",
            category: "docs",
            verbose: "Documentation",
            match: {
                message: ""
            }
        },
        {
            url: "https://10.1.16.99/docs/automation/playbooks",
            category: "docs",
            verbose: "Documentation",
            match: {
                message: ""
            }
        },
        .
        .
        .
    ]
}
```

The return values of note follow:

Field	Description
url	The url at which the matching search result is located
category	The category that the search result belongs to. Category could be one of the following: • container • artifact • asset • app • action • playbook • docs
match	The content that actually matched in the search result. This is a dictionary where the key is the field where the match was found. In Splunk Phantom versions before 4.0, the 'match' field used to return the search term used. In 4.0, it is empty.
verbose	Verbose description of the category in which the match was found.

Vault endpoints

REST Vault

/rest/vault_document

Get a list of vault documents.

Syntax

```
https://<username>:<password>@<host>/rest/vault_document
```
GET

Fetch a list of vault documents.

Example
A successful GET returns a list of vault documents.

```
{
    "count": 2,
    "num_pages": 1,
    "data": [
        {
            "tags": [],
            "id": 1,
            "contains": [
                "vault id"
            ],
            "first_seen_time": "2020-10-02T16:40:14.180537Z",
            "hash": "71b20ece47a85e9567225a1e25bfac827454f9ce",
            "meta": {
                "md5": "22b60c8b3bdf157772ccde6afd6d909a",
                "sha1": "71b20ece47a85e9567225a1e25bfac827454f9ce",
                "size": 42842,
                "sha256": "95c9a949ccb5e3b5337a4294df0830d5c12c7eec4995024f92153bd4e5a3c17c",
                "contains": [
                    "vault id"
                ]
            },
            "names": [
                "electric-plug.png"
            ],
            "sources": [
                {
                    "user_id": 3
                },
                {
                    "user_id": 3
                }
            ],
            "size": 42842,
            "version": 1,
            "deleted": false
        },
        {
            "tags": [],
            "id": 2,
            "contains": [
                "vault id"
            ],
            "first_seen_time": "2020-10-02T16:40:53.862877Z",
            "hash": "8526180d1f10a67de16b3f2bece163cb3c039cfd",
            "meta": {
                "md5": "82e658dff169edb23c29cdf6373d3e82",
                "sha1": "8526180d1f10a67de16b3f2bece163cb3c039cfd",
                "size": 484897,
                "sha256": "bfdc5bd10f344f4cb36c47f72a308deadbfa2ae4552f329509c0b6896a790fc8",
                "contains": [
                    "vault id"
                ]
            },
            "names": [
                "Screen Shot 2020-09-23 at 12.02.17 PM.png"
            ],
            "sources": [
                {
                    "user_id": 3
                }
            ],
            "size": 484897,
            "version": 1,
            "deleted": false
        }
    ]
}
```

Use the page filter to control how many documents you want to show at a time. For example, to see 10 records at a time use `page_size`.

Example request

```
/rest/vault-document?page_size=10
```

To choose which 10 records to display out of a list of 100, add `page` to the request:

Example request

```
//rest/vault_document?page_size=10&page=0
/rest/vault_document?page_size=10&page=1
```

/rest/vault_document/1

Fetch an individual vault document.

Syntax

```
https://<username>:<password>@<host>/rest/vault_document/1
```

GET

Fetch an individual vault document.

Example
A successful GET returns the individual vault document.

```
{
    "tags": [],
    "id": 1,
    "contains": [
        "vault id"
    ],
    "first_seen_time": "2020-10-02T16:40:14.180537Z",
    "hash": "71b20ece47a85e9567225a1e25bfac827454f9ce",
    "meta": {
        "md5": "22b60c8b3bdf157772ccde6afd6d909a",
        "sha1": "71b20ece47a85e9567225a1e25bfac827454f9ce",
        "size": 42842,
        "sha256": "95c9a949ccb5e3b5337a4294df0830d5c12c7eec4995024f92153bd4e5a3c17c",
        "contains": [
            "vault id"
        ]
    },
    "names": [
        "electric-plug.png"
    ],
    "sources": [
        {
            "user_id": 3
        },
        {
            "user_id": 3
        }
    ],
    "size": 42842,
    "version": 1,
    "deleted": false
}
```

Warm standby endpoints

REST Warm standby

API endpoints for getting warm standby information.

/warm_standby_check

To determine is a Splunk SOAR (On-premises) instance is part of a warm standby configuration an HTTP GET is made to `/warm_standby_check`.

Syntax

```
https://<username>:<password>@<host>/warm_standby_check
```
GET

Warm standby check

Example request
Check to see if an instance is configured as the standby in a warm standby pair.

```
curl -k -u admin:changeme https://localhost/warm_standby_check -G -X GET
```
Example response
A successful GET will return either 500 or 200.

- If the Splunk SOAR (On-premises) instance is the standby in a warm standby pair, the API will return `500`.
- If the Splunk SOAR (On-premises) instance is either the primary in warm standby pair, or if warm standby is not configured, the API will return `200`.

The API will return a result of 500 if used on a cluster node.

Splunk SOAR (On-premises) clusters cannot use the warm standby feature.

Workbook endpoints

REST Workbook

Manage workbooks or workflows.

A workbook is a response plan that has been added to a container, while a workbook template is the base object used to create workbooks. When a workbook template is added to a container, a copy of the template is added, which then becomes a workbook.

For information on adding a workbook template to a container, see REST Containers.

In Splunk Phantom 4.5, an alias 'workbook' was added for each 'workflow' REST API. For example /rest/workbook_template can be used interchangeably with /rest/workflow_template.

/rest/workbook_task filters

Get the tasks for all workbooks.

Syntax

```
https://<username>:<password>@<host>/rest/workbook_task
```

Usage details

By default, returns all tasks for all workbooks. You cannot filter results for a single workbook.

GET

Get workbook tasks belonging to a container.

Request parameters

Retrieve workbooks with the following parameters:

Field	Required	Type	Description
container_id	Optional	string	Identity of the container for the workbooks.
id	Optional	string	Identity of the workbook task.

Example request

Get the workbook tasks belonging to container 1.

```
curl -k -u admin:changeme https://localhost/rest/workbook_task?_filter_container_id=1&sort=id&order=desc -G
-X GET
```

Example response

A successful request returns a 200 response and a JSON object containing tasks.

```
{
    "count": 38,
    "num_pages": 4,
    "data": [
        {
            "id": 1,
            "name": "Confirmed incident",
            "order": 5,
            "owner": null,
            "role": null,
            "container": 1,
            "phase": 1,
            "description": "Confirmed incident - document the investigation and gather evidence. Attach all
relevant information from detection steps to the case",
            "suggestions": {},
            "status": 0,
            "create_time": "2022-10-19T22:15:41.046930Z",
            "modified_time": "2022-10-19T22:15:41.047202Z",
            "notified": false,
            "is_note_required": true,
            "sla": null,
            "sla_type": "minutes",
            "start_time": null,
            "end_time": null,
            "notes": [],
            "files": [],
            "owner_name": null
        },
        ...
    ]
  }
```

/rest/workbook_template

Create a workbook using `/rest/workflow_template` or `/rest/workbook_template`.

Syntax

```
https://<username>:<password>@<host>/rest/workbook_template
```

Usage details

The "Edit Container" permission is required to add or modify workbooks.

POST

Add a workbook template.

Request parameters

Workbook templates are modified with the following parameters.

Field	Required	Type	Description
is_default	required	Boolean	Sets the created workbook to be the default case workbook. Removes the default flag from any other workbook.
name	required	string	The name of the new workbook.
phases	optional	Array of JSON Objects	Contains an array of phase objects that describe the workflow. See /rest/workbook_phase_template for phase object parameters and /rest/workbook_task_template for task object parameters.

Example request

Create a workbook with the name "My Workbook" that includes a phase named "My Phase" and a task named "My Task."

```
curl -k -u admin:changeme https://localhost/rest/rest/workbook_template \
-d '{
        "name": "My Workbook",
        "is_default": false,
        "phases": [{
                "name": "My Phase",
                "order": 1,
                "tasks": [{
                        "name": "My Task",
                        "order": 1,
                        "description": "Investigate the event",
                        "playbooks": [{
                                        "scm": "local",
                                        "playbook": "investigate"
                                },
                                {
                                        "scm": "community",
                                        "playbook": "04_07_2017 - PhishMe"
                                }
                        ],
                        "actions": ["geolocate ip", "block_ip"]
                }...]
        }...]
}'
```

Example response

A successful request will result in a 200 response returning the new workbook Id and success as JSON.

```
{
    "id": 10,
    "success": true
}
```

/rest/workbook_phase_template

Add a phase to a workbook using `/rest/workbook_phase_template` or `/rest/workflow_phase_template`.

Syntax

```
https://<username>:<password>@<host>/rest/workbook_phase_template
```

Usage details

Be sure to include the `template_id` attribute.

POST

Add a phase to an existing workbook template.

Request parameters

A Phase Object describes the stages of life of a case and are composed of multiple tasks.

Field	Required	Type	Description
name	required	string	The name of the phase.
order	optional	integer	Specifies where the order the phase is expected to be executed in relative to other phases in the workbook.

Field	Required	Type	Description
tasks	optional	Array of JSON Objects	Contains an array of task objects that describe the phase. See /rest/workbook_task_template for the task object parameters.
template_id	optional	integer	May only be passed when adding a phase to an existing template.

Example request

Add a phase named "My Phase" and a task named "My Task" to template Id 10.

```
curl -k -u admin:changeme https://localhost/rest/rest/workbook_phase_template \
-d '{
  "name": "My Phase",
  "order": 1,
  "template_id": 10,
  "tasks": [
    {
      "name": "My Task",
      "order": 1,
      "description": "Investigate the event",
      "playbooks": [
        { "scm": "local", "playbook": "investigate" },
        { "scm": "community", "playbook": "04_07_2017 - PhishMe" }
      ],
      "actions": [ "geolocate ip", "block_ip" ]
    }...
  ]
}'
```

Example response

```
{
    "id": 20,
    "success": true
}
```

/rest/workbook_task_template

Add a task to an existing phase using `/rest/workbook_task_template` or `/rest/workflow_task_template`.

Syntax

```
https://<username>:<password>@<host>/rest/workbook_task_template
```

Usage details

Be sure to include the `phase_id` attribute.

POST

Add a task to an existing phase.

Request parameters

A Task Object indicates what activities should be taken on a case and may include playbooks, Splunk SOAR actions, or manual activities taken by a person.

Field	Required	Type	Description

Field	Required	Type	Description
actions	required	Array of strings	Array of action names that are expected to be carried out.
description	required	string	A textual description of the task to be performed.
playbooks	required	Array of JSON Objects	An array describing what playbook should be run. Each JSON object should contain 2 key-value pairs. They should have an "scm" key where the value is the name of the Source Control repository and a "playbook" key where the value is the name of the playbook within that repository.
phase_id	optional	integer	May only be passed when adding a task to an existing phase.
name	required	string	A human friendly name for the task.
order	optional	integer	Order in which the task is expected to be carried out relative to other tasks in the phase.
owner	optional	integer or string	The account name or numeric ID of the user or role who is the current owner of the container.

Example request

Add a task named "My Task" to existing phase Id 20.

```
curl -k -u admin:changeme https://localhost/rest/workbook_task_template \
-d '{
        "name": "My Task",
        "order": 1,
        "owner": 2,
        "phase_id": 20,
        "description": "Investigate the event",
        "playbooks": [{
                        "scm": "local",
                        "playbook": "investigate"
                },
                {
                        "scm": "community",
                        "playbook": "04_07_2017 - PhishMe"
                }
        ],
        "actions": ["geolocate ip", "block_ip"]
}'
```

Example response

A successful request will result in a 200 response returning the new task id and success as JSON.

```
{
    "id": 30,
    "success": true
}
```

Optional /rest/workbook_phase/ filters

Get the phases of a workbook.

/rest/workflow_phase?_filter_container_id

Syntax

```
https://<username>:<password>@<host>/rest/workbook_phase?_filter_container_id=<arguments>
```

GET

Get the phases of a workbook.

Example request
Get the phases of workbook Id 259.

```
curl -k -u admin:changeme https://localhost/rest/workflow_phase?_filter_container_id=259 -G -X GET
```

Example response
A successful request returns a 200 response and a JSON object containing the phases for the workbook.

```
{
  "count": 5,
  "data": [
      {
          "status": 0,
          "tasks": [
              {
                  "status": 0,
                  "files": [],
                  "owner_name": null,
                  "container": 259,
                  "name": "Determine if an incident has occurred",
                  "notified": false,
                  "start_time": null,
                  "suggestions": {},
                  "modified_time": "2019-05-30T19:06:24.687501Z",
                  "id": 5,
                  "phase": 1,
                  "create_time": "2019-05-30T19:06:24.687190Z",
                  "role": null,
                  "sla_type": "minutes",
                  "sla": null,
                  "owner": null,
                  "end_time": null,
                  "notes": [],
                  "order": 1,
                  "is_note_required": true,
                  "description": ""
              }, ...
          ],
          "container": 259,
          "name": "Detection",
          "start_time": null,
          "modified_time": "2019-05-30T19:06:24.658201Z",
          "id": 1,
          "create_time": "2019-05-30T19:06:24.657855Z",
          "sla_type": "minutes",
          "sla": null,
          "order": 1,
          "end_time": null
      }, ...
      {
          "status": 0,
          "tasks": [ ...
          ],
          "container": 259,
          "name": "Post Incident Activity",
            "start_time": "2019-05-30T19:06:25.670935Z",
            "modified_time": "2019-05-30T19:06:25.672107Z",
            "id": 5,
            "create_time": "2019-05-30T19:06:24.745617Z",
            "sla_type": "minutes",
            "sla": null,
            "order": 5,
            "end_time": null
        }
    ],
    "num_pages": 1
}
```

Multi-tenancy endpoints

REST Tenant

The tenant API allows you to get information about configured tenants on your Splunk SOAR (On-premises) deployment. Multi-tenancy is only supported for on premises deployments of Splunk SOAR.

/rest/tenant

Get information about the tenants configured on the Splunk SOAR (On-premises) deployment.

For information on configuring multi-tenancy, see Configure multiple tenants on your Splunk SOAR (On-premises) instance.

Syntax

```
https://<username>:<password>@<host>/rest/tenant
```

GET

Get a list of all configured tenants.

Example request
Get a list of tenants, using paging parameters.

```
curl -k -u admin:changeme https://localhost/rest/note?page_size=5&page=0 -G -X GET
```

Example response
A successful GET will return a 200 response, and a JSON formatted list of tenants. This example has multi-tenancy turned on, but only the default tenant is configured.

```
{
    "count": 1,
    "num_pages": 1,
    "data": [
        {
            "id": 0,
            "name": "Default",
            "description": "This is the default system tenant.",
            "contact": "contact name",
            "disabled": false,
            "slas": {
                "high": 60,
                "medium": 720,
                "low": 1440
            },
            "executive_approvers": null,
            "sla_expiry_warning_in_mins": null,
            "environment_variables": {}
        }
    ]
}
```