Vectra AI

Version 2.5 REST API Guide

VECTRA®

Revision History

DATE	COMMENT
October 2025	Release 9.6 New Endpoint for Active Directory (AD) Groups • /api/v2.5/settings/active_directory/groups Adds support for AD Groups in the following endpoints: • /api/v2.5/groups • /api/v2.5/groups/<id>
August 2025	Release 9.4 New Endpoint for Health Events • /api/v2.5/events/health
June 2025	Release 9.3 Limits the number of notes per entity returned in list endpoints. The most recent ten notes will be returned with the first truncated to 1000 characters and the remaining truncated to 100 characters. New Endpoints for closing/opening Detections, Hosts, Accounts • /api/v2.5/detections/close • /api/v2.5/detections/<id>/close • /api/v2.5/detections/open • /api/v2.5/detections/<id>/open • /api/v2.5/hosts/<id>/close • /api/v2.5/accounts/<id>/close Added new response field and query parameters • /api/v2.5/detections: ○ 'reason' parameter/field ○ 'include_src_dst_groups' query param ○ 'src_groups' and 'dst_groups' fields • /api/v2.5/detections/<id> ○ 'reason' parameter/field ○ 'include_src_dst_groups' query param ○ 'src_groups' and 'dst_groups' fields

DATE	COMMENT
April 2025	Release 9.2 • Adds support for getting group members in api/v2.5/groups/<id>/members endpoint. Release 9.1 • Adds support for group type migration for the /v2.5/groups/<id> endpoint. • Adds support for accessing the Vectra REST API via the api/v2.5/oauth2/token endpoint. • Adds new query parameter support for GET /v2.5/threatfeeds
March 2025	Release 9.0 • Adds support for regex groups in api/v2.5/groups and api/v2.5/groups/<id>
June 2024	Release 8.5 • Adds support for vectra-match/download-vectra-ruleset endpoint
March 2024	Release 8.4 • Adds the field ‘associated_accounts’ & marks field ‘subaccounts’ for deprecation in /api/v2.5/accounts & /api/v2.5/accounts/<id>
December 2023	Release 8.2 • Adds the query params “last_detection_timestamp_gte” & “last_detection_timestamp_lte” for Accounts and Hosts
September 2023	Release 8.2 • Adds the patch query parameter “membership_action” for Groups
May 2023	Release 7.6 • Adds support for /vectra-match/rules optional rotation flag on POST requests
Feb 2023	Release 7.5 • Adds support for all /vectra-match endpoints

DATE	COMMENT
Feb 2023	Release 7.5 • Removes support for the account_ids query parmater for the GET /groups endpoint

API Version 2.5 Changelog

OAuth2 Authentication

- Added /oauth2/token endpoint to support accessing the Vectra REST API using OAuth2 authentication

Detections

- Added the following params and fields for `v2.5/detections/`:
    - ‘reason’ parameter and field
- Added the following endpoints v2.5/detections/:
    - detections/close
    - detections/<detection_id>/close
    - detections/open
    - detectons/<detection_id>/open

Groups

- Removes support for the account_ids query parmater for the GET /groups endpoint
- Adds support for Dynamic Groups
- Adds support for group type migration, the ability to transition groups from static to dynamic and dynamic to static
- As part of our ongoing efforts to refine and improve our API, we have removed the `importance` field & query parameter. This field was introduced in error in a prior release but does not have a functional purpose in APIv2.
- Adds limitation to amount of members returned from groups in all methods for /groups and /groups/<id>. This is to improve performance around dynamic groups with high member counts.
- Added `v2.5/settings/active_directory/groups` endpoint
- AD Groups support now available in paths `v2.5/groups` and `v2.5/groups/<id>`

Group Members

- Adds support for getting group members by group ID through the `/groups/<id>/members` endpoint.
- Adds ability to filter group members by name. Host group members may be filtered by name or is_key_asset.

Vectra Match

- Adds support for all `/vectra-match` endpoints

Accounts

- ‘subaccounts’ renamed to ‘associated_accounts’. ‘subaccounts’ will be deprecated in v2.6
- ‘email,’ in the ‘ldap’ object renamed to ‘email’. ‘email,’ will be deprecated in v2.6
- Added `v2.5/accounts/<account_id>/close` endpoint

Health

- Added `/health/detection` endpoint

Hosts

- Added `v2.5/accounts/<account_id>/close` endpoint

Health Events

- Added `/api/v2.5/events/health` endpoint to query health events

Overview

A REST API is available for administrators and developers to integrate Vectra's breach detection data into their applications. Vectra X-series RESTful API provides access to security event data, platform configuration, and health information via URI paths.

Vectra REST API is based on open standards. You can use any web development language to access and retrieve information via the API. A common use-case would be to retrieve security event information generated by the Vectra platform and supply this information to a security operations dashboard or incident response and ticketing systems.

The REST API can be accessed via HTTPS connection to the management interface IP address of the Vectra X-series. The data in the response to the API query is in JSON format.

Examples of security event data that can be integrated into your application:

- Security event type detected
- Host/Account information associated with the security event
- Severity of the Host/Account activities

The Vectra REST API is accessible using token authentication or OAuth2 authentication. A token can be generated on the 'My Profile' page from the Vectra Dashboard.

Security Detection Data

The "Detections", "Hosts", and "Accounts" elements retrieve security events that can be inserted into external applications. The REST API provides filtering options to extract data. Advanced parsing of the data can be performed after data has been retrieved and saved into your target application. Order of the response data returned is latest first.

Accessing REST API 2.5

The REST API v 2.5 is accessed via the URL `https://<vectra-management-ip>/api/v2.5/<path>`.

The `<path>` options for REST API queries are listed in the table below.

URL	METHOD	DATA TO QUERY
`/accounts`	`GET`	Account
`/accounts/<account_id>/close`	`PATCH`	Close single Account
`/assignments`	`GET, POST, PUT, DELETE`	Host and account assignments
`/assignment_outcomes`	`GET, POST, PUT, DELETE`	Host and account assignment outcomes
`/audits`	`GET`	Audit logs
`/campaigns`	`GET`	Campaigns
`/detections`	`GET, PATCH`	Security detection events
`/detections/close`	`PATCH`	Close multiple detections
`/detections/open`	`PATCH`	Open multiple detections
`/detections/<detection_id>/close`	`PATCH`	Close single detection
`/detections/<detection_id>/open`	`PATCH`	Open single detection
`/groups`	`GET, POST`	Groups
`/groups/<id>`	`GET, PATCH, DELETE`	Group
`/groups/<id>/members`	`GET`	Group Members
`/health`	`GET`	System Health information
`/hosts`	`GET, PATCH`	Host information
`/hosts/<host_id>/close`	`PATCH`	Close single Host
`/ip_addresses`	`GET`	IP information related to host sessions
`/lockdown/account`	`GET`	Accounts disabled via Lockdown
`/lockdown/host`	`GET`	Hosts disabled via Lockdown
`/proxies`	`GET, POST, PATCH`	Proxy objects
`/rules`	`GET, POST, PUT, DELETE`	Triage rule configuration
`/search`	`GET`	Advanced search on hosts, accounts and detections
`/sensor_token`	`GET, POST, DELETE`	Brain sensor registration token

URL	METHOD	DATA TO QUERY
`/settings`	`GET, POST`	HostID external connectors for AWS
`/settings/active_directory/groups`	`GET`	Active Directory Groups from connected Active Directories
`/subnets`	`GET`	Get all subnets associated with known hosts
`/tagging`	`GET, PATCH`	Tags for hosts and detections
`/threatFeeds`	`GET, POST, DELETE`	threatFeed objects
`/threatFeeds/<id>`	`GET, POST, DELETE, PATCH`	threatFeed metadata
`/threatFeeds/<id>/file`	`GET, DELETE`	threatFeed file contents
`/traffic`	`GET`	Get all network traffic stats
`/usage/detect`	`GET`	Get count of concurrent ips on detect network
`/users`	`GET, PATCH`	User objects
`/vectra-match/enablement`	`GET, POST`	Enable/disable and view whether Vectra Match is enabled on a given paired device (WARNING: POSTing to the enablement endpoint causes a reboot if changing state)
`/vectra-match/stats`	`GET`	Vectra Match stats
`/vectra-match/status`	`GET`	Vectra Match status
`/vectra-match/available-devices`	`GET`	List of currently available devices on which Vectra Match can be enabled
`/vectra-match/rules`	`GET, POST, DELETE`	Upload and manage Vectra Match rulesets
`/vectra-match/assignment`	`GET, POST, DELETE`	Manage Vectra Match assignments (which rules are associated with which devices)
`/vectra-match/alert-stats`	`GET`	View the most common alerts that Vectra Match has generated
`/Vectra-match/download-vectra-ruleset`	`GET`	Download the Vectra curated ruleset file
`/vsensor`	`GET`	Download Vsensor information
`/events/health`	`GET`	Health Events

The REST API is available for all Vectra admin user roles. Version 2.5 uses token authentication or OAuth2 Authentication.

Token Authentication

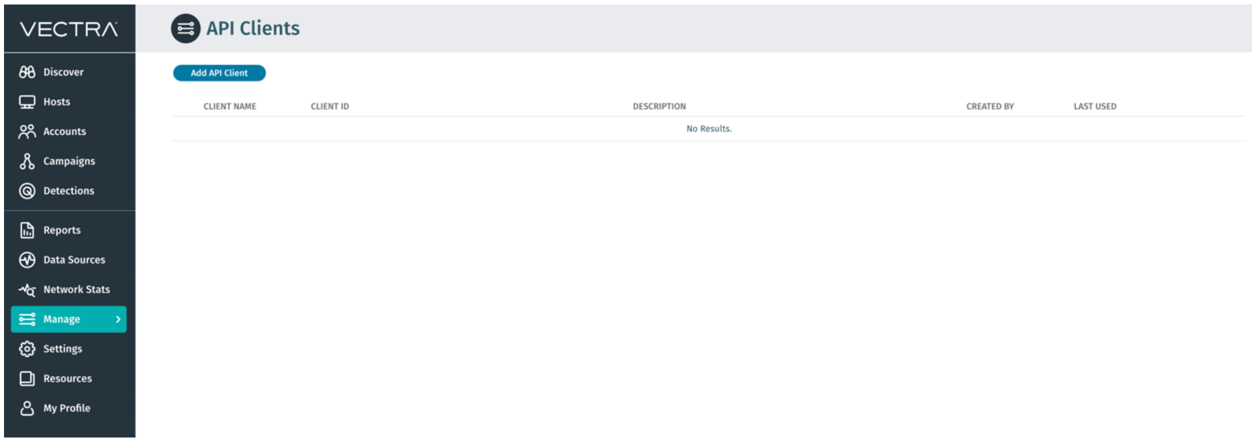
Every user created on the Vectra platform has access to his or her API token under the "My Profile" page. The token provides access to the API based on the user's role, like UI access. Thus, if a user does not have ability to create triage filters from the UI, that user will also not be able to create triage filters using API.

Example of using token authentication with curl is shown below.

```
curl –H "Authorization: Token db20f83b33744690e4168e7994c8dd0b53e64f94"
https://<vectra-management-ip>/api/v2.5/
```

API Clients

Getting access to the Vectra Platform API can be done through the creation of an API Client. Creation of an API Client will provide a set of OAuth 2.0 credentials that will be used to gain authorization to the Vectra Platform API. Please note that management of API Clients is restricted to Detect users with the role of "Super Admin". To create an API client, log into your Detect portal and navigate to *Manage* > *API Clients*.

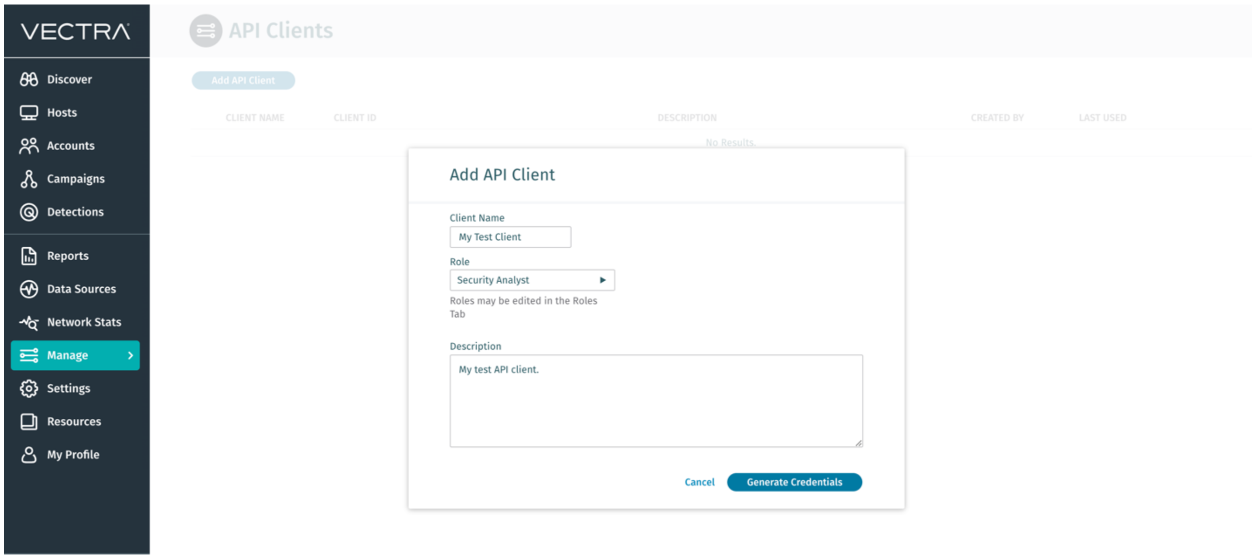


CLIENT NAME	CLIENT ID	DESCRIPTION	CREATED BY	LAST USED

No Results.

Creating a new API Client

From the API Clients page, select 'Add API Client' to create a new client.



Add API Client

Client Name: My Test Client

Role: Security Analyst

Roles may be edited in the Roles Tab

Description: My test API client.

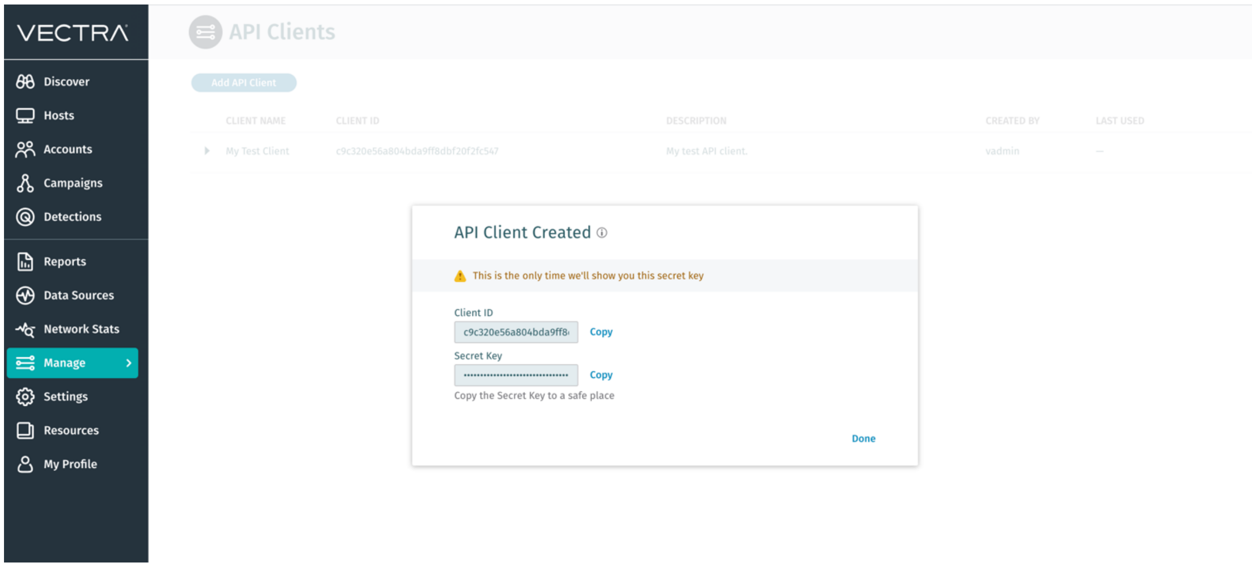
Creating a new API Client has two required parameters:

- **Name –** a user-friendly name to identify the client (up to 256 characters)
- **Role –** the role maps the API Client to a set of permissions, similar to the way a Detect UI user would be assigned a role. The role must be one of the following:
    - Read-Only
    - Restricted Admin
    - Security Analyst
    - Settings Admin
    - Auditor

Creating a new API Client has one optional parameter:

- **Description** – a brief description to aid in identifying the client (up to 2048 characters)

Once you have entered the API Client information, select ‘Generate Credentials’ to obtain your client credentials.



CLIENT NAME	CLIENT ID	DESCRIPTION	CREATED BY	LAST USED
My Test Client	c9c320e56a804bda9ff8dbf20f2fc547	My test API client.	vadmin	—

API Client Created

This is the only time we'll show you this secret key

Client ID: c9c320e56a804bda9ff8

Secret Key: ••••••••••••••••••••••••••••••

Copy the Secret Key to a safe place

Be sure to record your **Client ID** and **Secret Key** for safekeeping. You will need these two pieces of information to obtain an access token from the Vectra Platform API. An access token is required to make requests to all of the Vectra Platform API endpoints.

Authenticating as an API Client

Many REST API tools and libraries can programmatically manage access tokens for you once provided the OAuth 2.0 parameters described below, or you can manually make requests to obtain them as needed.

First you must authenticate using the **Client ID** and **Secret Key** you obtained when creating your API Client.

Example Request:

```
POST https://<vectra_portal_url>/api/v2.5/oauth2/token
Authorization: Basic <HTTPBasic(<client_id>:<secret_key>)>
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
```

Example Response:

```
{
    "access_token": "Z0FBQUFB...",
    "expires_in": 21600,
    "token_type": "Bearer"
}
```

Notice that the response includes an **access_token** which will be used to make subsequent API requests as a client, outlined in the next sections. An access_token will expire in 6 hours and is expressed in seconds. Once your access_token has expired, you will need to reauthenticate your API client with your client_id and secret_key in order to receive a new access_token.

Requests via Postman

If you are using Postman, we have a public API collection you can import to your environment and use.

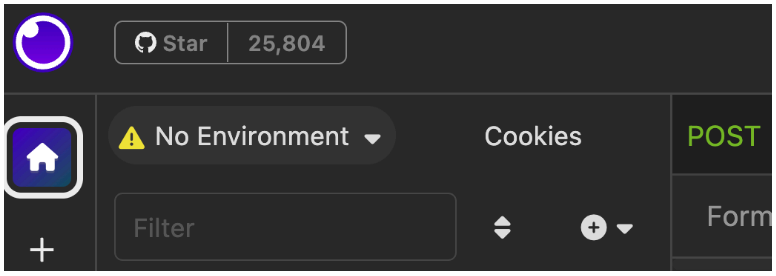
Please refer to the Quickstart guides here:

1. REST API Quick Start Guide for Postman v2.5 Using OAuth2 (QUX)
2. REST API Quick Start Guide for Postman v2.5 Using Token Auth (QUX)

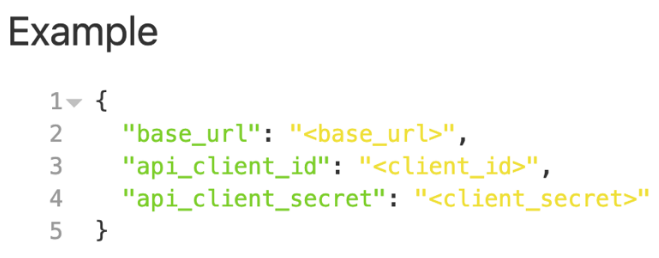
Requests via Insomnia

If you are using Insomnia for API requests, we recommend the following setup for basic Authentication requests:

2. Create a new environment with your API client credentials. Select the "No Environment" dropdown found on the top left corner of Insomnia, choose "Manage Environments".



3. Use the example data below to create a new sub environment. Give the environment a unique name.

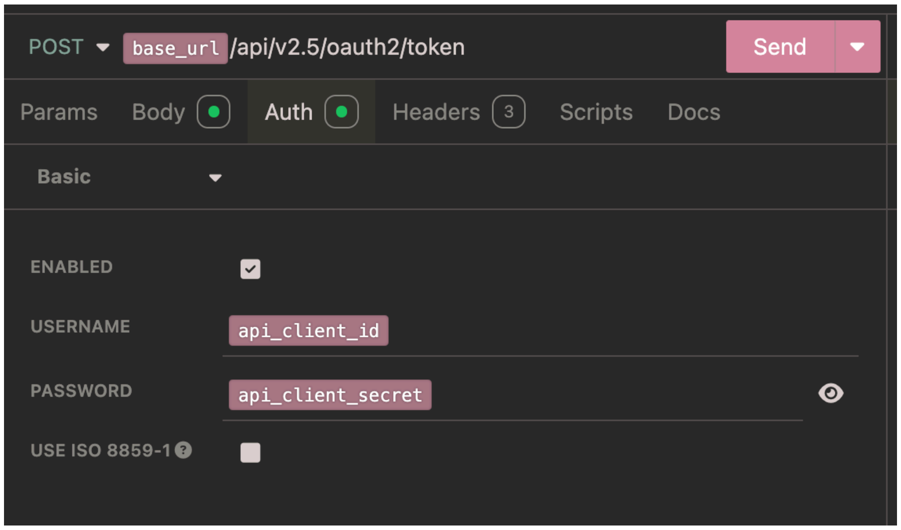


Example

```
{
  "base_url": "<base_url>",
  "api_client_id": "<client_id>",
  "api_client_secret": "<client_secret>"
}
```

4. Repeat for any other desired tenants.
5. Select your desired environment to start making requests.

6. Create a new request for /oauth2/token & set up basic authentication. Reference the client_id and secret from your environment file via Insomnia variables in the Basic auth tab (example below).



POST base_url /api/v2.5/oauth2/token

Basic

ENABLED ✓

USERNAME api_client_id

PASSWORD api_client_secret

USE ISO 8859-1 ☐

<u>Example Request</u>:

```
POST https://<vectra_portal_url>/api/v2.5/oauth2/token
Authorization: Basic <HTTPBasic(client_id:secret_key)>
Content-Type: application/x-www-form-urlencoded
```

<u>Example Response</u>:

```
{
    "access_token": "Z0FBQUFB...",
    "expires_in": 21600,
    "token_type": "Bearer"
}
```

Make API request as a client

Now that you have obtained an **access_token**, you will use that token to make API requests as an API Client to all the Vectra Platform API endpoints. Here are a couple quick examples.

<u>Example Request to accounts endpoint</u>:

```
GET /api/v2.5/accounts
Authorization: Bearer <access_token>
```

<u>Example Response</u>:

```
{
    "count": 16,
    "next": "http://<vectra_portal_url>/api/v2.5/accounts?page=1",
    "previous": null,
    "results": [
    ...
}
```

Example Request to detections endpoint:

```
GET /api/v2.5/detections
Authorization: Bearer <access_token>
```

Example Response:

```
{
    "count": 25,
    "next": "http://<vectra_portal_url>/api/2.5/detections?page=2",
    "previous": null,
    "results": [
    ...
}
```

Full sample output for accounts and detections endpoints can be found in Appendix A.

Triage

Version 2.5 of triage API supports GET, POST, PUT, and DELETE. The API endpoint for accessing version 2.5 is `https://<vectra-management-ip>/api/v2.5/rules`

The following table lists the top-level fields and descriptions present in the response for a bulk GET request to the v2.5 API for Triage rules.

ELEMENT	DESCRIPTION	TYPE	NOTES
count	The number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	Useful when using a web-based REST API browser.
previous	URL link to the previous page of output	string	Useful when using a web-based REST API browser.
results	A list of all Triage rules (described in the table below) being returned by the query	list	

The following table lists the fields and descriptions present in a Triage rule. These Triage rules will be contained inside the ‘results’ field, which is a top-level field described in the table above.

ELEMENT	DESCRIPTION	TYPE	NOTES
id	The ID of the Triage rule	integer	
url	A v2.5 URL to the Triage rule	string	Useful when using a web-based REST API browser.
enabled	Describes if the Triage rule is enabled	Boolean	
created_timestamp	Describes the time the Triage rule was created	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
last_timestamp	The timestamp when this Triage filter was triggered	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
is_whitelist	This whitelists all detections for this activity	Boolean	True or False
priority	Used in ordering execution of Triage filters	integer	
active_detections	The total number of active detections this Triage rule applies to	integer	
total_detections	The total number of detections (active or inactive) this Triage rule applies to	integer	
template	Specifies if this Triage rule was created based off of a template	Boolean	True of False
detection_category	Original detection category	string	
triage_category	Custom Triage label used to categorize specified detections	string	
detection	Original detection type	string	
source_conditions	Specifies the entity this Triage rule applies to	JSON	source_conditions represents the conditions that can be applied to the source of a detection, including host, account, IP, and sensor. For more information on the format of source_conditions, see below this table

ELEMENT	DESCRIPTION	TYPE	NOTES
additional_conditions	Specifies additional matching criteria for this Triage rule	JSON	additional_conditions are other conditions that are different on a per-detection-type basis. For more information on the format of additional_conditions, see below this table

Formatting source_conditions and additional_conditions

The format of version 2.5 Triage rules is based on AND/OR logic, making them more flexible and customizable to your specific situation. Both ‘source_conditions’ and ‘additional_conditions’ are now saved as JSON blobs which represents a tree-like structure. Each tree node is represented in the following format:

```
{operator: operand}
```

Where “operator” is any of the following:

```
For non-leaf nodes: ‘AND’ or ‘OR’
For leaf nodes: ‘ANY_OF’ or ‘NONE_OF’
```

And “operand” is any of the following:

```
For non-leaf nodes: A list of children tree nodes of the form [{operator:
                    operand}, {operator: operand}, …]
                  For leaf nodes: A dictionary of the form:
                              {
                                    ‘field’: <FIELD NAME>,
                                    ‘values’: [
                                          {‘value’: [<VALUE>]}
                                    ],
                                    ‘group’: [
                                          {‘value’: [<GROUP ID>]}
                                    ]
                              }
```

Stipulations on the format of the top-level tree structure as of 2.5 is that the top-level operator must be an ‘OR’ node, with a single ‘AND’ node as the only child. The ‘AND’ node may have an arbitrary number of leaf node children. All valid 2.5 Triage rules will look as follows:

```
{
      ‘OR’: [
            { ‘AND’: [
                  <LEAF NODE 1>,
                  <LEAF NODE 2>,
                  …
                  <LEAF NODE N>,
            ]}
      ]
}
```

source_conditions and additional_conditions have different fields that can be used in their leaf nodes. For source_conditions, the following fields are valid: ip, host, account, sensor. The fields ‘ip’ and ‘host’ also support triaging on groups, meaning that an ip or a host leaf node can be given an IP Group or Host Group ID, and the Triage rule will apply to every IP/host in the specified group.

For additional_conditions, the following fields are valid: remote1_ip, remote1_proto, remote1_port, remote1_dns, remote2_ip, remote2_proto, remote2_port, remote2_dns, account, named_pipe, uuid, identity, share, extensions, rdp client name, rdp client token, keyboard name. The fields ‘remote1_ip’, ‘remote2_ip’, ‘remote1_dns’, ‘remote2_dns’ support triaging on groups.

Either source_conditions or additional_conditions may be null. Setting source_conditions to null implies that this Triage rule with apply to All Hosts.

To see examples of complete valid source_conditions and additional_conditions, see Appendix A.

Detections

Detection objects contain all the information related to security events detected on the network. The URL to retrieve all detections is `https://<vectra-management-ip>/api/v2.5/detections` and uses Token auth.

Version 2.5 introduces listing/adding/updating/removing ‘Notes’ on a Detection entity via GET, POST, PATCH, DELETE methods. The API endpoint for accessing this in version 2.5 is:
`https://<vectra_management_ip>/api/v2.5/detections/<detection_id>/notes`
See: Appendix A: ‘Detections (Notes)’ section for examples.

Detections are grouped into one of the following categories:

DETECTION CATEGORY	URL
Command & Control	`/detections?category=command`
Botnet	`/detections?category=botnet`

DETECTION CATEGORY	URL
Reconnaissance	`/detections?category=reconnaissance`
Lateral Movement	`/detections?category=lateral`
Exfiltration	`/detections?category=exfiltration`
Info	`/detections?category=info`

The API query performs partial word match. For example, you can use `/detections?category=recon` to query all Reconnaissance category detections.

An example of using curl to retrieve all detections using token authentication:

```
curl –H “Authorization: Token db20f83b33744690e4168e7994c8dd0b53e64f94”
https://<vectra-management-ip>/api/v2.5/detections
```

The following table lists the top-level fields and descriptions present in the response for a bulk GET request to the v2.5 API for Detections

ELEMENT	DESCRIPTION	TYPE	NOTES
count	The number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	Useful when using a web-based REST API browser.
previous	URL link to the previous page of output	string	Useful when using a web-based REST API browser.
results	The list of Detections returned in the output	List of Objects	

The following table lists the fields and descriptions present in a Detection. These Detections will be contained inside the ‘results’ field, which is a top-level field described in the table above.

ELEMENT	DESCRIPTION	TYPE	NOTES
id	Object ID	integer	
detection	The name of the threat detected.	string	See Appendix B for the list of Detection names
detection_type	The name of the threat detected.	string	See Appendix B for the list of Detection names

ELEMENT	DESCRIPTION	TYPE	NOTES
category	The category of the vname attack detected.	string	See Appendix B for the list of categories. Will be deprecated in future release. Replaced by detection_category.
detection_category	The category of the vname attack detected.	string	See Appendix B for the list of categories.
src_ip	The source IP address of the host attributed to the security event.	string	
state	The state of the detection.	string	If the detection has aged out the state will transition to “active” from “inactive”. If marked as fixed in the UI, or via the V2.5 API, state will be “fixed”.
t_score	The threat score attributed to the detection.	integer	Will be deprecated in future release. Replaced by threat.
threat	The threat score attributed to the detection.	integer	
c_score	The certainty score attributed to the detection.	integer	Will be deprecated in future release. Replaced by certainty.
certainty	The certainty score attributed to the detection.	Integer	
created_timestamp	Describes the time the Triage rule was created	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
first_timestamp	The timestamp when the event was first detected	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
last_timestamp	The timestamp when the event was last detected	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
description	System generated description of the event.	string	
proto	Protocol used in the communications.	string	

ELEMENT	DESCRIPTION	TYPE	NOTES
total_bytes_sent	Total bytes sent by the client.	integer	
total_bytes_rcvd	Total bytes received by the client.	integer	
url	The URL that links directly to this record.	string	
sensor_name	The name of sensor where this flow was detected from.	string	
src_host	A dictionary with fields that describe the Host the detection is from	JSON object	
url	The URL that links directly to the detection record	string	
summary	The summary information for the detection	JSON object	See Appendix A for examples
grouped_details	The detection details for the detection	JSON object	See Appendix A for examples
tags	User defined tags added to the detection	List of strings	
targets_key_asset	Indicates whether the detection targets a key asset	Boolean	Will be deprecated in future release. Use is_targeting_key_asset.
campaign_summaries	The summaries of campaigns of which this detection is part.	JSON object.	The summaries of campaigns this detection belongs to
is_targeting_key_asset	Indicates whether the detection targets a key asset	Boolean	
note	User defined note for this detection.	string	

ELEMENT	DESCRIPTION	TYPE	NOTES
note_modified_by	Username who last modified note	string	
note_modifed_timestamp	The timestamp when note was last modified	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
notes	An array of the most recent ten notes on the detection	array of objects	Includes id, date_created, date_modified, created_by, modified_by, and note contents. Notes after the first will be truncated to 100 characters.
assigned_to	User named assigned to this detection	string	
assigned_date	The timestamp when user was assigned this detection	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
src_account	A dictionary with fields that describe the Account the detection is from	JSON object	
reason	Reason the detection was closed	String	
src_groups	Groups related to the source of the detection	Array of objects	Only included in response if \`include_src_dst_groups\` param is set to one of (True, true, 1) in the request
dst_groups	Groups related to the destination of the detection	Array of objects	Only included in response if \`include_src_dst_groups\` param is set to one of (True, true, 1) in the request

You can also apply filters to the API response to query for specific elements. The available filter options can be viewed by using OPTIONS instead of GET within the REST client.

The available options and filters for detection set is listed below.

QUERY PARAMETER	DESCRIPTION
fields	Filters objects listed. The available fields are listed in Table 3
page	Page number. Possible values are a positive integer or last
page_size	Page size. Possible values are a positive integer, up to 5000.

QUERY PARAMETER	DESCRIPTION
ordering	Orders records by last timestamp, threat score and certainty score. The default sorts threat and certainty score in ascending order. Scores can be sorted in descending order by prepending the query with “minus” symbol.
min_id	>= the id provided
max_id	<= the id provided
state	filter by state: active, inactive, ignored, ignored for all
category	filter by the detection category
detection_type	filter by the name of the threat detected.
detection_category	filter by the detection category
src_ip	filter by source (ip address)
t_score	filter by threat score
t_score_gte	filter by threat score >= the score provided
threat_score	filter by threat score
threat_gte	filter by threat score >= the score provided
c_score	filter by certainty score
c_score_gte	filter by certainty score >= the score provided
certainty	filter by certainty score
certainty_gte	filter by certainty score >= the score provided
last_timestamp_gte	filter by last_timestamp >= the date provided
last_timestamp	filter by last timestamp
host_id	filter by id of the host object a detection is attributed to
tags	filter by a tag or a comma-separated list of tags
destination	filter by destination in the detection detail set
proto	filter by the protocol in the detection detail set
is_targeting_key_asset	filter by is_targeting_key_asset: True or False
note_modified_timestamp_gte	filter by note_modified_timestamp >= the timestamp provided: ‘2019-08-27T20:55:29Z’
reason	filter by reason: benign, remediated
include_src_dst_groups	Include ‘src_groups’ and ‘dst_groups’ fields in response.

Examples of detection queries:

QUERY	COMMENT
`https://1.1.1.2/api/v2.5/detections/?ordering=t_score`	Retrieves all detections with threat score sorted low to high (ascending).
`https://1.1.1.2/api/v2.5/detections/?ordering=-t_score`	Retrieves all detections with threat score sorted high to low (descending).
`https://1.1.1.2/api/v2.5/detections/?c_score_gte=60`	Retrieves all detections with certainty score greater than or equal to 60.
`https://1.1.1.2/api/v2.5/detections/?page=2`	Retrieves page 2 of detections.
`https://1.1.1.2/api/v2.5/detections/?t_score_gte=90&c_score_gte=90`	Retrieves all detections with threat &certainty score greater than or equal to 90
`https://1.1.1.2/api/v2.5/detections/?tags=RAT`	Retrieves all detections with tag value=RAT
`https://1.1.1.2/api/v2.5/detections/?tags=RAT,TOR`	Retrieves all detections with tag value=RAT, and TOR

Example using a Python script to retrieve “Detections” event data of type “Data Smuggler” and from host 172.16.106.116.

```
import requests
import json
vectra_url = 'https://192.168.51.13/api/v2.5/detections'
headers = {'Content-Type': 'application/json', 'Authorization': 'Token
053662afeb2d02bd3773b93bbc514d4bbb92694d'}
payload = {'detection': 'data smuggler', 'src_ip': '172.16.106.116'}
response = requests.get(url=vectra_url, params=payload, verify=False,
headers=headers)
print(response.json())
```

Detection PCAP API

Version 2.0 of the API supports download of PCAP for a particular detection. Users can perform a GET using the following URL format to download pcap for any detection:

```
https://<vectra-management-ip>/api/v2.5/detections/<id>/pcap
```

The download of pcap via API is subject to RBAC rules just like UI access. If a user’s role does not permit viewing pcap information, he or she will not be able to access pcap via API as well.

Close Detections

Closing a detection indicates that some remedial action was taken based upon the detection.

Closing a detection will no longer contribute to the scoring.

Closing detections requires the following elements to be present

- detectionIdList (Only applies to detections/close)
- reason

URL to close detections is

`https://<vectra_portal_url>/api/v2.5/detections/close`
`https://<vectra_portal_url>/api/v2.5/detections/<detection_id>/close`

Open Detections

Re-opening a detection will result detection being rescored.

Re-opening detections requires the following element to be present

- detectionIdList (Only applies to detections/open)

URL to close detections is

`https://<vectra_portal_url>/api/v2.5/detections/open`
`https://<vectra_portal_url>/api/v2.5/detections/<detection_id>/open`

Account

Version 2.5 of accounts API supports GET. The API endpoint for accessing version 2.5 is
`https://<vectra_management_ip>/api/v2.5/accounts`

Version 2.5 introduces listing/adding/updating/removing ‘Notes’ on an Account entity via GET, POST, PATCH, DELETE methods. The API endpoint for accessing this in version 2.5 is:
`https://<vectra_management_ip>/api/v2.5/accounts/<account_id>/notes`
See: Appendix A: ‘Account (Notes)’ section for examples.

The following table lists the top-level fields and descriptions present in the response for a bulk GET request to the v2.5 API for Accounts

ELEMENT	DESCRIPTION	TYPE	NOTES
count	The number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	Useful when using a web-based REST API browser.
previous	URL to the previous page of output	string	Useful when using a web-based REST API browser.

ELEMENT	DESCRIPTION	TYPE	NOTES
results	A list of all Accounts (described in the table below) being returned by the query	list	

The following table lists the fields and descriptions present in an Account. These Accounts will be contained inside the 'results' field, which is a top-level field described in the table above.

ELEMENT	DESCRIPTION	TYPE	NOTES
id	The ID of the Account	integer	
url	A v2.5 URL to the Account	string	Useful when using a web-based REST API browser.
name	The name associated with the Account	string	
state	The state of the Account	string	
threat	The threat score attributed to the account	integer	
certainty	The certainty score attributed to the account	Integer	
severity	The severity of this Account	string	Either 'Low', 'Medium', 'High', or 'Critical'. This is calculated based off of the threat and certainty of the Account
account_type	The method through which this account was discovered	List of strings	Can be one or both of "kerberos" or "o365"
tags	User defined tags added to the account	List of strings	
sensors	Sensors associated with the accounts detection set	List of strings	
note	User defined note added to the account	string	
note_modified_by	Username who last modified note	string	
note_modifed_timestamp	The timestamp when note was last modified	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT

ELEMENT	DESCRIPTION	TYPE	NOTES
notes	An array of the first ten notes on the account	array of objects	Includes id, date_created, date_modified, created_by, modified_by, and note contents. Notes after the first will be truncated to 100 characters.
privilege_level	A number 1-10 to represent how privileged this account is	int	
privilege_category	A string to represent how privileged this account it	string	Either 'Low', 'Medium', or 'High'. Privilege levels of 1 and 2 map to 'Low'. Privilege levels of 3-7 map to 'Medium'. Privilege levels of 8-10 map to 'High'
last_detection_timestamp	Last detection activity from this Account	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
detection_set	List of Detections for Account	string	
detection_summaries	The summaries of detections attached to this Account.	List of Objects	
ldap	Information about the LDAP server this Account came from (if applicable)	JSON object	'email,' in the 'ldap' object to be deprecated in favor of 'email' in v2.6
subaccounts	List of associated accounts	List of strings	To be deprecated in favor of 'associated_accounts' in v2.6
associated_accounts	List of associated accounts	List of strings	

The available options and filters for account set is listed below.

QUERY PARAMETER	DESCRIPTION
fields	Filters objects listed
page	Page number. Possible values are a positive integer or last
page_size	Page size. Possible values are a positive integer up to 5000.
ordering	Orders records by last timestamp, threat score and certainty score. The default out sorts threat and certainty score in ascending order. Scores can be sorted in descending order by prepending the query with "minus" symbol.
name	filter by name
state	filter by state: active or inactive
t_score	filter by threat score

QUERY PARAMETER	DESCRIPTION
t_score_gte	filter by threat score >= the score provided
c_score	filter by certainty score
c_score_gte	filter by certainty score >= the score provided
tags	filter by a tag or a comma-separated list of tags (returns hosts that contain any of the tags specified), e.g.tags=baz | tags=foo,bar"
all	No filter, return all host objects. Only available in version 2.0 API
min_id	Filter hosts have id greater than or equal to min_id
max_id	Filter hosts have id less than or equal to max_id
note_modified_timestamp_gte	filter by note_modified_timestamp >= the timestamp provided: ‘2019-08-27T20:55:29Z’
privilege_level	filter by exact privilege level of hosts. 1-10
privilege_level_gte	filter hosts that have a privilege level greater than or equal to the supplied number. 1-10
privilege_category	filter hosts by privilege category. Options are ‘low’, ‘medium’ and ‘high’
last_detection_timestamp_gte	filter by last_detection_timestamp >= timestamp provided
last_detection_timestamp_lte	filter by last_detection_timestamp <= timestamp provided

Accounts Close

Closing an account indicates that some remedial action was taken based upon the account.

Active detections associated with the account will be closed resulting in the detections no longer contributing to the scoring

Closing an account requires the following elements to be present

- reason

URL to close an account is

`https://<vectra_portal_url>/api/v2.5/accounts/<account_id>/close`

Campaigns

Version 2.5 of campaigns API supports GET. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/campaigns`

The following table lists the top-level fields and descriptions present in the response for a bulk GET request to the v2.5 API for Campaigns.

ELEMENT	DESCRIPTION	TYPE	NOTES
count	The number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	Useful when using a web-based REST API browser.
previous	URL to the previous page of output	string	Useful when using a web-based REST API browser.
results	A list of all Campaigns (described in the table below) being returned by the query	list	

The following table lists the fields and descriptions present in Campaigns. These Campaigns will be contained inside the 'results' field, which is a top-level field described in the table above.

ELEMENT	DESCRIPTION	TYPE	NOTES
id	The ID of the Campaign	integer	
dst_ip	IP address where Campain originated	string	
`target_domain`	Domain attacker is targeting	string	
`state`	Current state of the Campaign	string	"Active" or "Inactive"
`name`	Name associated with the Campaign	string	
`last_modified`	Time Campaign was last updated	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
`note`	Note information	object	
`note_modified_by`	User who last modified the Note	string	
`note_modified_timestamp`	Time Notes was last updated	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT

The available options and filters for account set is listed below.

QUERY PARAMETER	DESCRIPTION
fields	Filters objects listed
dst_ip	Filter by known destination ip address

QUERY PARAMETER	DESCRIPTION
target_domain	Filter by target domain
state	Filter by either active or inactive Campaigns
name	Filter by name
last_updated_gte	Filter Campaigns that have been updated after date specified
note_modified_timestamp_gte	Filter Notes that have been updated after date specified

Hosts

Host information includes data that correlates the Host data to detected security events. This information includes but is not limited to:

- Hostname
- IP address
- Threat score
- Certainty score

URL to retrieve hosts information is `https://<vectra_management_ip>/api/v2.5/hosts`.

Example using a Python script to retrieve "Hosts" information from hostname "TB5-7" is shown below using token authentication. To retrieve the most current host information, you must issue the request with a "state=active" for the current host information. The host information retrieved will include active detection data attributed to that host. A similar search could have been made against the IP address of the host, using the last_source parameter.

```
import requests
import json
vectra_url = 'https://192.168.22.34/api/v2.5/hosts'
headers = {'Content-Type': 'application/json', 'Authorization': 'Token 
053662afeb2d02bd3773b93bbc514d4bbb92694d'}
payload = {'state': 'active', ‘name’: ‘TB5-7’}

response = requests.get(url=vectra_url, params=payload, verify=False, 
headers=headers)
print(response.json())
```

An example of using curl to retrieve all hosts using token authentication:

```
curl –H “Authorization: Token db20f83b33744690e4168e7994c8dd0b53e64f94”
https://<vectra_management_ip>/api/v2.5/hosts
```

The hosts API using token authentication will return hosts that have active detections, active traffic or are marked as key assets. To return all the hosts, use query parameter "all"

Version 2.5 introduces listing/adding/updating/removing 'Notes' on a Host entity via GET, POST, PATCH, DELETE methods. The API endpoint for accessing this in version 2.5 is:
`https://<vectra_management_ip>/api/v2.5/hosts/<host_id>/notes`
See: Appendix A: 'Host (Notes)' section for examples.

The following table lists fields and description of the various elements for the hosts.

ELEMENT	DESCRIPTION	TYPE	NOTES
count	The number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	Useful when using a web-based REST API browser.
previous	URL to the previous page of output	string	Useful when using a web-based REST API browser.
results	List of hosts returned in the output	List of Objects	

The following table lists the fields and descriptions present in a Host. These Hosts will be contained inside the 'results' field, which is a top-level field described in the table above.

ELEMENT	DESCRIPTION	TYPE	NOTES
id	Object ID	integer	
new_host_pointer	Indicates whether additional records for this host exists	string	Deprecated in version 2.0 of API
name	The learned hostname	string	
state	The state of this host	string	Possible values are "active" or "inactive". Active hosts are those with active detections
active_traffic	Flag indicating if the host is exhibiting traffic	Boolean	Only used in version 2.0 of host API. Possible values are True or False. Will be deprecated in future release. Replaced by has_active_traffic.
has_active_traffic	Flag indicating if the host is exhibiting traffic	Boolean	Only used in version 2.0 of host API. Possible values are True or False.
t_score	The current threat score correlated to this host	integer	Will be deprecated in future release. Replaced by threat.
threat	The current threat score correlated to this host	integer	

ELEMENT	DESCRIPTION	TYPE	NOTES
c_score	The current certainty score correlated to this host	integer	Will be deprecated in future release. Replaced by certainty.
certainty	The current certainty score correlated to this host	integer	
last_source	Last source IP associated with this host	string	
previous_ips	The previous ips observed for this host	List of strings	Only used in version 2.0 of API
last_detection_timestamp	Last detection activity from this host	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
key_asset	Flag indicating if the host is a key asset	Boolean	Only used in version 2.0 of API. Possible values “True” or “False”. Will be deprecated in future release. Replaced by is_key_asset.
is_key_asset	Flag indicating if the host is a key asset	Boolean	Only used in version 2.0 of API. Possible values “True” or “False”.
targets_key_asset	Flag indicating if the host has a detection targeting a key asset	Boolean	Only used in version 2.0 of API. Possible values “True” or “False”. Will be deprecated in future release. Replaced by is_targetin_key_asset.
is_targeting_key_asset	Flag indicating if the host has a detection targeting a key asset	Boolean	Only used in version 2.0 of API. Possible values “True” or “False”.
probable_owner	Probable owner of the host	string	
tags	User defined tags added to the host	List of strings	
note	User defined note added to the host	string	
note_modified_by	Username who last modified note	string	
note_modifed_timestamp	The timestamp when note was last modified	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
notes	An array of the first ten notes on the host.	array of objects	Includes id, date_created, date_modified, created_by, modified_by, and note contents. Notes after the first will be truncated to 100 characters.
host_artifact_set	List of host artifacts observed for the host	List of Objects	

ELEMENT	DESCRIPTION	TYPE	NOTES
sensor	Sensor that reported seeing the activity for this host	string	
detection_set	List of Detections for Host	string	
url	The URL to link directly to this host record	string	
detection_summaries	The summaries of detections attached to this host.	List of Objects	
campaign_summaries	The summaries of campaigns of which this host is part.	List of Objects	
assigned_to	User named assigned to this detection	string	
assigned_date	The timestamp when user was assigned this detection	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
privilege_level	A number 1-10 to represent how privileged this account is	int	
privilege_category	A string to represent how privileged this account it	string	Either 'Low', 'Medium', or 'High'. Privilege levels of 1 and 2 map to 'Low'. Privilege levels of 3-7 map to 'Medium'. Privilege levels of 8-10 map to 'High'

The available options and filters for hosts set is listed below.

QUERY PARAMETER	DESCRIPTION
fields	Filters objects listed
page	Page number. Possible values are a positive integer or last
page_size	Page size. Possible values are a positive integer up to 5000.
ordering	Orders records by last timestamp, threat score and certainty score. The default out sorts threat and certainty score in ascending order. Scores can be sorted in descending order by prepending the query with "minus" symbol.
name	filter by name
state	filter by state: active or inactive
last_source	filter by last_source (ip address)
t_score	filter by threat score

QUERY PARAMETER	DESCRIPTION
t_score_gte	filter by threat score >= the score provided
c_score	filter by certainty score
c_score_gte	filter by certainty score >= the score provided
last_detection_timestamp	filter by last_detection_timestamp
last_detection_timestamp_gte	filter by last_detection_timestamp >= timestamp provided
last_detection_timestamp_lte	filter by last_detection_timestamp <= timestamp provided
tags	filter by a tag or a comma-separated list of tags (returns hosts that contain any of the tags specified), e.g.tags=baz | tags=foo,bar"
key_asset	filter by key asset: True, False
targets_key_asset	filter by hosts targeting key assets: True, False. Only available in version 2.0 API
all	No filter, return all host objects. Only available in version 2.0 API
active_traffic	Filter by hosts that have generated active traffic within the last 2 hours. Only available in version 2.0 API
min_id	Filter hosts have id greater than or equal to min_id
max_id	Filter hosts have id less than or equal to max_id
mac_address	filter by mac address
note_modified_timestamp_gte	filter by note_modified_timestamp >= the timestamp provided: '2019-08-27T20:55:29Z'
privilege_level	filter by exact privilege level of hosts. 1-10
privilege_level_gte	filter hosts that have a privilege level greater than or equal to the supplied number. 1-10
privilege_category	filter hosts by privilege category. Options are 'low', 'medium' and 'high'

Hosts Close

Closing a host indicates that some remedial action was taken based upon the host.

Active detections associated with the host will be closed resulting in the detections no longer contributing to the scoring

Closing a host requires the following elements to be present

- reason

URL to close a host is

`https://<vectra_portal_url>/api/v2.5/hosts/<host_id>/close`

Search

The search API endpoint allows users to perform advanced search against hosts and detections. All attributes of hosts and detections as described in the above sections are searchable (only the ones exposed in the version 2.0 of the endpoints).

Hosts

The search endpoint for hosts is:

`https://<vectra_management_ip>/api/v2.5/search/hosts/?page_size=<number>&&query_string=<query>`

where <query> is the query that needs to be performed.

page_size is optional. If specified, the page_size will specify the number of results returned per page. If not specified, the default page_size is 50. The maximum value for page_size is 5000.

Some example queries on hosts:

QUERY	DESCRIPTION
`‘host.threat:>=50 and host.certainty:>=50’`	Find all hosts in the critical quadrant
`‘host.suspicious_admin_learnings.host_manages. protocols:ssh’`	Find hosts that administer other hosts over ssh
`‘host.suspicious_admin_learnings.managers_of_host. protocols:vnc’`	Find hosts that are administered over vnc
`‘host.tags:"alice"’`	Find hosts which match the exact tag “alice”
`‘host.notes:test’`	Find notes that contains the phrase “test”
`‘host.detection_summaries.summary.dst_ports:389’`	Find hosts that have detections that target port 389
`‘host.last_detection_timestamp:’\[now-1d TO now\]’`	Find hosts that have had a detection in the last 24 hours

Accounts

`The search endpoint for hosts is:`

`https://<vectra_management_ip>/api/v2.5/search/accounts/?page_size=<number>&&query_string=<query>`

where <query> is the query that needs to be performed.

page_size is optional. If specified, the page_size will specify the number of results returned per page. If not specified, the default page_size is 50. The maximum value for page_size is 5000.

Some example queries on accounts:

QUERY	DESCRIPTION
`‘account.threat:>=50 and account.certainty:>=50’`	Find all Accounts in the critical quadrant
`‘account.tags:"alice"’`	Find Accounts which match the exact tag “alice”
`‘account.notes:test’`	Find notes that contains the phrase “test”
`‘account.detection_summaries.summary.dst_ports:389’`	Find Accounts that have detections that target port 389
`‘acount.last_detection_timestamp:\[now-1d TO now\]’`	Find Accounts that have had a detection in the last 24 hours

Detections

The search endpoint for detections is:

`https://<vectra_management_ip>/api/v2.5/search/detections/?page_size=<number>&query_string=<query>`

where <query> is the query that needs to be performed.

page_size is optional. If specified, the page_size will specify the number of results returned per page. If not specified, the default page_size is 50. The maximum value for page_size is 5000.

Some example queries on detections:

QUERY	DESCRIPTION
`‘detection.is_triaged:false and detection.grouped_details.target_domains:snakeoil.biz’`	Find all active detections that have a target domain of ‘snakeoil.biz’
`‘detection.grouped_details.dst_ports:445’`	Find all detections on dst_port 445
`‘detection.grouped_details.shares:test’`	Find all detections that have targeted share “test”
`‘detection.grouped_details.user_agent:Safari’`	Find all detections that have observed user_agent “Safari”
`‘detection.custom_detection:Misconfiguration’`	Find all triaged detections with name “Misconfiguration”

QUERY	DESCRIPTION
`‘detection.last_timestamp:\[now-1d to now\] and (detection.is_triaged:false)’`	Find all active detections that had detection activity in the last day

Users

User information includes all data corresponding to user accounts. This information includes but is not limited to:

URL to retrieve users information is `https://<vectra_management_ip>/api/v2.5/users/`

Example using a Python script to retrieve “Users” information from username “cognito” is shown below using token authentication.

```
import requests
import json
vectra_url = 'https://192.168.51.13/api/v2.5/users'
headers = {'Content-Type': 'application/json', 'Authorization': 'Token 
053662afeb2d02bd3773b93bbc514d4bbb92694d'}
payload = {'username': 'vadmin'}
response = requests.get(url=vectra_url, params=payload, verify=False, 
headers=headers)
print(response.json())
```

An example of using curl to retrieve all hosts using token authentication:

```
curl –H “Authorization: Token db20f83b33744690e4168e7994c8dd0b53e64f94” 
https://<vectra_management_ip>/api/v2.5/users
```

The following table lists fields and description of the various elements for the Users.

ELEMENT	DESCRIPTION	TYPE	NOTES
count	Number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	Useful when using a web-based REST API browser.
previous	URL link to the previous page of output	string	Useful when using a web-based REST API browser.
results	List of users returned in the output	List of Objects	

The following table lists the fields and descriptions present in a User. These Users will be contained inside the 'results' field, which is a top-level field described in the table above.

ELEMENT	DESCRIPTION	TYPE	NOTES
id	Object ID	integer	
last_login	Last time the user logged in	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
username	Account name	string	
email	Email associated with the account	string	
account_type	Account type	string	Local, Special, Limited Time Link, LDAP, TACACS
authentication_profile	Name of the User's authentication profile	string	LDAP or TACACS only
role	User's group role	string	

The available options and filters for Users is listed below.

QUERY PARAMETER	DESCRIPTION
username	Filter by username
role	Filter by role
account_type	Filter by account type
authentication_profile	Filter by authentication profile
last_login_gte	Filters for User's that have logged in since the given timestamp

Examples of user queries

QUERY	COMMENT
`https://1.1.1.2/api/v2.5/users/?page=2`	Retrieves page 2 of users
`https://1.1.1.2/api/v2.5/users/?username=cognito`	Retrieves all users with the username Cognito
`https://1.1.1.2/api/v2.5/users/?role=admin`	Retrieves all users with the role of admin
`https://1.1.1.2/api/v2.5/users/?account_type=local`	Retrieves all users with the account type of local
`https://1.1.1.2/api/v2.5/users/?authentication_profile=auth-profile`	Retrieves all users with either LDAP or TACACS profiles' names auth-profile

QUERY	COMMENT
`https://1.1.1.2/api/v2.5/users/?last_login_gte='2019-08-27T20:55:29Z'`	Retrieves all users that have logged in since 2019-08-27T20:55:29Z

ThreatFeeds

The threatFeeds API can be used to automate the upload of STIX files for threat intelligence matching. The API endpoint can also be used to retrieve the current list of threatFeed objects already configured in the system

To obtain the list of threatFeeds, use the following URL
`https://<vectra_management_ip>/api/v2.5/threatFeeds`

To obtain details of any threat feed, use the GET method on following URL
`https://<vectra_management_ip>/api/v2.5/threatFeeds/<id>`

To delete the threat feed, use the DELETE method on the following URL
`https://<vectra_management_ip>/api/v2.5/threatFeeds/<id>`

where <id> is the id of the threatFeed.

The following table lists the fields and a description of the various elements for the "threatFeeds". Detailed examples of using GET, POST on threatFeeds is described in Appendix A

ELEMENT	DESCRIPTION	TYPE	NOTES
name	The name of the threat feed	string	
duration	The default duration for which indicators in the threat feed are valid	integer	If the indicator in the STIX file has an expiry date, that will be used. The default duration is used for indicators that don't have any expiry in the STIX file. Every time the file is updated, the default expiry will be the upload date + duration specified.
category	The category in which the detection will fire if a match is observed with any indicator in the threatFeed	string	Valid values are "cnc", "lateral" or "exfil"
indicatorType	The default indicatorType to use for the observables in the STIX file	string	Valid values are "Anonymization", "C2", "Exfiltration", "Malware Artifacts", "Watchlist". The value specified will be used for observables that don't have an indicator associated with them in the STIX file

ELEMENT	DESCRIPTION	TYPE	NOTES
certainty	The default certainty to use for indicators in the STIX file	string	Valid values are “Low”, “Medium” or “High”. The certainty value specified will be used if indicator in the STIX file doesn’t have a certainty associated with it.
data	The data from the STIX file	Json object	If the threatFeed has a STIX file already uploaded, then a GET will return the contents of the STIX file in the “data” object, otherwise the data object will be null

The available query parameters to filter for Threat Feeds are listed below. NOTE: These filters will only apply to Threat Feeds with an associated stix file.

QUERY PARAMETER	TYPE	DESCRIPTION
category	string	Category of the upload
certainty	string	Certainty of the stix file
expiration	string ISO8601	Expiration date of the stix file (exact match)
expiration_gte	string ISO8601	Expiration date of the stix file is greater than or equal to date provided
expiration_lte	string ISO8601	Expiration date of the stix file is less than or equal to date provided
last_updated	string ISO8601	Last updated date of the stix file (exact match)
last_updated_gte	string ISO8601	Last updated date of the stix file is greater than or equal to date provided
last_updated_lte	string ISO8601	Last updated date of the stix file is less than or equal to date provided
last_updated_by	string	Name of user who last modified the Threat Feed
name	string	Name of the Threat Feed

Proxies

The proxies API can be used to manage proxy IP addresses (internal or external) in Cognito. The API can be used to retrieve the current list of proxy IP addresses or to create new proxy objects in Cognito.

To obtain the list of proxies use the following URL
https://<vectra_management_ip>/api/v2.5/proxies/

To obtain details of any proxy, use the following URL
`https://<vectra_management_ip>/api/v2.5/proxies/<id>`

where <id> is the id of the proxy object.

The following table lists the fields and a description of the various elements for “proxies”. Detailed examples of using GET, POST, PATCH on proxies is described in Appendix A

ELEMENT	DESCRIPTION	TYPE	NOTES
source	Whether the proxy was auto detected by Cognito or was added by user	string	Valid values: “user” or “cognito”
id	The identifier of the proxy object	string	
considersProxy	Whether to consider the object as a proxy or not	string	Proxies auto detected by Cognito which have source value as “Cognito” will always have this field set to “true”. User defined proxies can be “true” or “false”
address	The proxy IP address	string	The IP address for the proxy object

Tagging

The tagging API can be used to manage tags for host and detections in Cognito. The API can be used to retrieve or update the current list of tags for either a host or detection.

To manage tags for a host, use the following URL
https://<vectra_management_ip>/api/v2.5/tagging/host/<id>

where <id> is the id of the host object.

To manage tags for a detection, use the following URL
https://<vectra_management_ip>/api/v2.5/tagging/detection/<id>

where <id> is the id of the detection object.

The following table lists the fields and a description of the various elements for tagging operations. Detailed examples of using GET, PATCH on proxies is described in Appendix A

ELEMENT	DESCRIPTION	TYPE	NOTES
status	Status of Tagging request.	string	Valid values: ‘success’ or ‘failure’
tags	List of tags for the host or detection object.	list of strings	When doing a PATCH operation, the object will be updated to match the list of tags provided. To remove tags for a host or detection set tags to an empty list.
message	Error message if operation was unsuccessful.	string	Only present when status is ‘failure’ of operation is a failure.

ELEMENT	DESCRIPTION	TYPE	NOTES
Invalid_tags	List of tags which are invalid for the host or detetion object.	list of strings	Only present when status is 'failure'. This will only be returned in the response for a PATCH operation.

Groups

The groups API can retrieve a listing of groups that are defined on the system.

Version 2.5 of groups API supports GET, PATCH, POST, and DELETE to not only query the list of groups, but also create, modify or delete them. The path for accessing groups API in version 2.5 of the API endpoint is `https://<vectra_management_ip>/api/v2.5/groups`

Note that dynamic groups with large amounts of users may cause the endpoint to time out. To avoid this, please set the "include_members" query param to False.

For version 2.5 and beyond, by default, 2000 members will be returned per group for all methods against /groups and /groups/<id>. The introduction of Dynamic Groups has increased the potential for groups with member counts into the thousands. Limiting the amount of members returned per group will result in overall improved performance of the Groups endpoint.
To access a list of all members for a group, please use the Group Members endpoint: v2.5/groups/<id>/members.

Examples of creating, modifying or viewing groups using API are also described in Appendix A.

A list of groups can be retrieved using the path:

`https://<vectra_management_ip>/api/v2.5/groups`

Example curl command for fetching a list of groups:

```
curl –H “Authorization: Token db20f83b33744690e4168e7994c8dd0b53e64f94”
      https://<vectra_management_ip>/api/v2.5/groups
```

Example curl command for creating a group:

```
curl -X POST –H “Authorization: Token
db20f83b33744690e4168e7994c8dd0b53e64f94”
      https://<vectra_management_ip>/api/v2.5/groups
      -d ‘{“name”: “New Group Name”,
           “description”: “New Group Description”,
           “type”: “account”,
           “members”: [1, 2]
```

```
    }'
```

The following table lists the fields and a description of the various elements for “groups”.

ELEMENT	DESCRIPTION	TYPE	NOTES
id	Object id	integer	Always present
name	The name of the group	string	Always present
type	The type of the group (e.g. host)	string	Always present
cognito_managed	Indication of whether group is managed by Cognito	boolean	Only present when type is “domain” or “ip”
description	User-defined description of the group	string	Always present
last_modified_timestamp	The datetime of the last modification to the group	datetime	Always present
last_modified_by	The user that last modified the group	string	Always present
members	List of member objects	array	Always present. Limited to 2000 members before truncation.
rules	List of triage rules that the group is attached to	array	Always present
regex	User defined regular expression used to match and add members to group.	string	
membership_evaluation_ongoing	True if background membership evaluation is ongoing.	Boolean	List of members will be empty until evaluation is complete.
member_count	The amount of members included in the group.		
built_using	Describes whether the group was made with “static_members” or “regex”, or “AD”		
members_truncated	True if the ‘members’ field has been truncated at 2000 members. False if all members have been returned.	Boolean	
ad_group_dn	Distinguished Name of the AD group	string	

The available query parameters to filter for Groups are listed below.

QUERY PARAMETER	DESCRIPTION
account_names	Only valid when the type parameter is set to “account”. Provide a comma-delimited (,) list of strings to filter groups associated with accounts having one of the given names.
domains	Only valid when the type parameter is set to “domain”. Provide a comma-delimited (,) list of strings to filter groups associated with doamins having one of the given domains.
host_ids	Only valid when the type parameter is set to “host”. Provide a comma-delimited (,) list of integers to filter groups associated with hosts having one of the given ids.
host_names	Only valid when the type parameter is set to “host”. Provide a comma-delimited (,) list of strings to filter groups associated with hosts having one of the given names.
ips	Only valid when the type parameter is set to “ip”. Provide a comma-delimited (,) list of strings to filter groups associated with ips having one of the given ips.
description	Filter by group description (case insensitive match).
last_modified_timestamp	Filters for all groups modified on or after the given timestamp (GTE), formatted in ISO-8601.
last_modified_by	Filters groups by the user who made the most recent modification.
name	Filters by group name (case insensitive match).
type	Filter by group type. This version of the API will only accept the values “account”, “host”, “domain”, and “ip”.
membership_action	Exclusive parameter for PATCH requests. This will accept the values “append”, “remove”, and “replace”. Members passed in the request body will be used to perform the corresponding membership update. Appendix A covers this in more detail.
is_regex	Filters groups by presence of a regular expression statement. Query will only include groups made with regex when True and groups without regex when False.
is_membership_evaluation_ongoing	Filters groups depending on if membership evaluation is currently ongoing.
include_members	Returns groups with a list of respective members if True, returns groups with ‘members’ field as an empty list if False. Dynamic groups may have hundreds or thousands of members and can cause the endpoint to timeout if this query param is not set to False.
ad_group_dn	Filter on AD group distinguished names.
is_ad_group	Filter on AD groups. Query will include groups made with AD when True and groups without AD when False.

Group Members

The Group Members API can retrieve a list of all members belonging to a Host, Account, Domain, or IP Group.

Version 2.5 of groups API supports GET to query the list of members belonging to the group of the respective ID.

The path for accessing the Group Members API in version 2.5 of the API endpoint is:
`https://<vectra_management_ip>/api/v2.5/groups/<id>/members`

The following table lists the top-level fields and descriptions present in the response for a bulk GET request to the v2.5 API for Group Members.

ELEMENT	DESCRIPTION	TYPE	NOTES
results	A list of member objects.	list	
next	URL to the next page of groups.	string	
previous	URL to the previous page of groups.	string	
count	The total number of members matching the query parameters.	integer	

The following tables list the fields and descriptions present in Member objects. The object differs for each type: host, account, domain, and ip.

Host Member:

Element	Description	Type	Notes
id	Host ID.	integer	
name	The Host name.	string	
is_key_asset	Whether or not the Host is a key_asset.	Boolean	
url	An API URL linking to the Host entity.	string	

Account Member:

Element	Description	Type	Notes
`uid`	Account unique ID.	integer	‘name’ can be used to filter this field

Domain Member:

Element	Description	Type	Notes
ip	IP member belonging to group.	string	‘name’ can be used to filter this field

IP Member:

Element	Description	Type	Notes
domain	Domain member belonging to group.	string	‘name’ can be used to filter this field

The available query parameters to filter for Group Members are listed below.

QUERY PARAMETER	TYPE	DESCRIPTION
page	integer	Page number. Possible values are a positive integer or last
page_size	integer	Page size. Possible values are a positive integer up to 5000.
`ordering`	string	Ordering for hosts members can be either ‘id’ or ‘name’
name	string	Filter by members that contain the string passed to the ‘name’ parameter. Will filter on different columns for different type: hosts – name accounts – uid ip – ip domain - domain
is_key_asset	Boolean	Filter host members by whether they are key assets.

Active Directory Groups

The Active Directory Groups API can retrieve a listing of Active Directory Distinguished Names from connected Active Directories.

The URL to retrieve active directory groups is
`https://<vectra_portal_url>/api/v2.5/settings/active_directory/groups`

The following table lists the top-level fields and descriptions present in the response for a bulk GET request to the v2.5 API for Active Directory Groups.

ELEMENT	DESCRIPTION	TYPE	NOTES
count	The number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	Useful when using a web-based REST API browser.
previous	URL to the previous page of output	string	Useful when using a web-based REST API browser.
results	A list of all Active Directory Groups from connected Active Directories (described in the table below) being returned by the query	list	

The following table lists the fields and descriptions present in an Active Directory Group object.

Element	Description	Type	Notes
ad_profile_name	The name of the connected AD profile	string	
dn	Distinguished Name	string	
existing_host_group	True if this AD group is imported as a Vectra AD host group	Boolean	
existing_account_group	True if this AD group is imported as a Vectra AD account group	Boolean	

The available query parameters to filter for Active Directory Groups are listed below.

QUERY PARAMETER	TYPE	DESCRIPTION
ad_profile_name	string	The name of the connected AD profile

An example of using the GET method for Active Directory Groups can be found in Appendix A.

Health

The health API can retrieve a listing of system health information.

Version 2.5 of health API supports GET. The path for accessing health API in version 2.5 of the API endpoint is `https://<vectra_management_ip>/api/v2.5/health`

Please note that the health API requires the view health permission to access the system health data via API.

Examples of viewing system health information using the API are also described in Appendix A.

A list of health information can be retrieved using the path:

`https://<vectra_management_ip>/api/v2.5/health`

Example curl command for fetching a list of system health information:

```
curl –H “Authorization: Token db20f83b33744690e4168e7994c8dd0b53e64f94”
      https://<vectra_management_ip>/api/v2.5/health
```

The following table lists the fields and a description of the various elements for “health”.

ELEMENT		DESCRIPTION	TYPE	NOTES
network		Network information	object	Updated every 5 minutes
	interfaces	A list of network interfaces on the Brain and Sensors	list of objects	Link speed shown in megabits per second
	traffic	Peak traffic values on Brain and Sensors	list of objects	Link speed shown in megabits per second
	vlan_count	Number of vlans observed	integer	
system		System information	object	
	serial_number	Serial number of Brain	string	
	uptime	Time since last power on	string	
	version	System version	object	Includes date of last update, version, mode, and model
memory		System memory	object	
	free_bytes	Memory available for system use	integer	Shown in bytes
	dimm_status	DIMM status and location information	list of objects	
	used_bytes	Memory in use	integer	Shown in bytes
	usage_percentage	Percentage of memory in use	integer	
	total_bytes	Total physical memory available	integer	Shown in bytes

ELEMENT		DESCRIPTION	TYPE	NOTES
sensors		Sensor information	list of objects	
	status	Pairing status of sensor	string	
	name	Name of the sensor	string	
	serial_number	Serial number of sensor	string	
	package_version	Sensor package version	string	
	last_seen	Timestamp of last seen heartbeat from Sensor	datetime	
	ip_address	IP address of Sensor	string	
	luid	Vectra-unique identifier for Sensor	string	
	location	Location of sensor	string	
disk		Disk information	object	
	raid_disks_missing	All disks in the RAID are present	object	Optional health status and errors, if any
	disk_utliization	Filesystem utilization	object	Includes free, used, total, and usage percentage of filesystem in bytes
	degraded_raid_volume	All healthy disks are being utilized by RAID	object	Optional health status and errors, if any
	disk_raid	Overall RAID health	object	Optional health status and errors, if any
cpu		Processor information	object	
	idle_percent	% of CPU idle	integer	
	user_percent	% of CPU processing tasks	integer	
	nice_percent	% of CPU processing higher prioritized tasks	integer	
	system_percent	% of CPU processing system specific tasks	integer	
hostid		HostID coverage health	object	

ELEMENT		DESCRIPTION	TYPE	NOTES
	ip_always	Number of IP addresses that are always covered by Host ID when seen on the network	integer	
	ip_sometimes	Number of IP addresses that are sometimes covered by Host ID when seen on the network	integer	
	ip_never	Number of IP addresses that are never covered by Host ID when seen on the network	integer	
	artifact_counts	The number of each type of artifact seen on the network	list of objects	Calculated weekly
power		Power supply information	object	
	status	Status of power supply	string	
	power_supplies	List of power supplies and temperature information	list of objects	Shown in Celsius, and optional faults, if any
	error	Power supply health errors		Optional power supply errors, if any
connectivity				
	sensors	List of sensors	array	Shown for each paired sensor
	last_check	The hour when the information for that check was curated	string	
	affected_metadata_hours	A list of hours during which the metadata pipeline had issues	string	Field not present if status is 'OK'
	error	A message detailing what issues happened in the metadata pipeline	string	
	status	The status of the sensor	string	Status can be OK, WARNING, CRITICAL, or UNKNOWN
	name	The name of the sensor	string	
	serial_number	The serial number of the sensor	string	

ELEMENT		DESCRIPTION	TYPE	NOTES
	luid	The Vectra-generated unique ID of the sensor	string	
	ip_address	The IP address of the sensor	string	
trafficdrop				Sensor output is only shown if traffic has gone out of baseline
	sensors	List of sensors	array	Shown for each paired sensor
	name	The name of the sensor	string	
	ip_address	The IP address of the sensor	string	
	luid	The Vectra-generated unique ID of the sensor	string	
	error	A message detailing the sensor status	string	Under normal operation this will show either “All interfaces have traffic volume within range” or “At least one interface had low traffic volume.”
	status	The status of the sensor	string	Status can be OK, WARNING, UNKNOWN, or SKIP
	start	The beginning of time the drop traffic was observed	string	
	end	The end of time the drop traffic was observed	string	
	interfaces	The interfaces that were observed out of baseline	array	
	name	The name of the interface	string	
	baseline	Provides the current calculated beaseline value (in packets) for the given interface	integer	
	cutoff	The value below which Vectra considers the interface to be out of baseline	integer	

ELEMENT		DESCRIPTION	TYPE	NOTES
	traffic	The observed traffic volume (in packets) for this interface	integer	
detection		Detection model health	object	
	check_results		array	One entry per failing detection model, or exactly one entry if all detection models are healthy
	name	The name of the check	string	
	detection_type	The name of the detection model	string	
	message	A message detailing the detection model health status	string	
	status	The status of the detection model	string	Status can be OK or CRITICAL

The available options and filters for Health is listed below.

QUERY PARAMETER	DESCRIPTION
cache	True by default. If the response is from cache, it will include an *updated_at* property with the datetime at which the health check was performed. Cached values will be no older than five minutes. Setting cache=false will force a new health check.
vlans	True by default. Setting vlans=false will omit the list of VLANs observed from the returned data. Large environments with many vlans can result in slow performance from the /health and /health/network endpoints.

Account Lockdown

The lockdown API can retrieve a listing of accounts that have been disabled in Active Directory via Detect's Lockdown function.

Version 2.5 of the lockdown API supports GET to query the list of disabled accounts. The path for accessing lockdown in version 2.5 of the API endpoint is `https://<vectra_management_ip>/api/v2.5/lockdown/account`.

An example of viewing disabled accounts using the API are also described in Appendix A.

A list of disabled accounts can be retrieved using the path:

https://<vectra_management_ip>/api/v2.5/lockdown/account

Example curl command for fetching a list of disabled accounts:

```
curl –H “Authorization: Token db20f83b33744690e4168e7994c8dd0b53e64f94”
      https://<vectra_management_ip>/api/v2.5/lockdown/account
```

The following table lists the fields and a description of the various elements for “lockdown”.

ELEMENT	DESCRIPTION	TYPE	NOTES
lock_date	The datetime the account was disabled	datetime	
locked_by	The name user that disabled the account	string	
unlock_date	The datetime the account is to be re-enabled per the configured Lockdown timer	datetime	
account_id	The ID of the Account	integer	
account_name	The name of the Account	string	

Host Lockdown

The lockdown API can retrieve a listing of hosts that have been disabled in Microsoft Defender ATP via Detect's Lockdown function.

Version 2.5 of the lockdown API supports GET to query the list of disabled hosts. The path for accessing lockdown in version 2.5 of the API endpoint is https://<vectra_management_ip>/api/v2.5/lockdown/host.

An example of viewing disabled hosts using the API are also described in Appendix A.

A list of disabled hosts can be retrieved using the path:

https://<vectra_management_ip>/api/v2.5/lockdown/host

Example curl command for fetching a list of disabled hosts:

```
curl –H “Authorization: Token db20f83b33744690e4168e7994c8dd0b53e64f94”
      https://<vectra_management_ip>/api/v2.5/lockdown/host
```

The following table lists the fields and a description of the various elements for “lockdown”.

ELEMENT	DESCRIPTION	TYPE	NOTES
lock_date	The datetime the host was disabled	datetime	
locked_by	The name user that disabled the host	string	
unlock_date	The datetime the host is to be re-enabled per the configured Lockdown timer	datetime	
host_id	The ID of the Host	integer	
host_name	The name of the Host	string	

Subnets

Version 2.5 of subnets API supports GET. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/subnets`

The following table lists the top-level fields and descriptions present in the response for a bulk GET request to the v2.5 API for Subnets.

ELEMENT	DESCRIPTION	TYPE	NOTES
count	The number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	Useful when using a web-based REST API browser.
previous	URL to the previous page of output	string	Useful when using a web-based REST API browser.
results	A list of all Subnets (described in the table below) being returned by the query	list	

The following table lists the fields and descriptions present in Subnets. These Subnets will be contained inside the ‘results’ field, which is a top-level field described in the table above.

ELEMENT	DESCRIPTION	TYPE	NOTES
subnet	IP address of the Subnet	string	
hosts	Hosts associated with the Subnet	string	

ELEMENT	DESCRIPTION	TYPE	NOTES
last_seen	Time when Subnet was last active	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
first_seen	Time when Subnet was last active	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT

The available options and filters for subnet set is listed below.

QUERY PARAMETER	DESCRIPTION
subnet	Filters objects listed
hosts	Filter by associated host
last_seen	Filter by time Subnet was last seen
first_seen	Filter by time Subnet was first seen

Syslog/Kafka configuration

Remote Syslog and Kafka servers can be configured, not created, through the API. Once a remote server has been configured through the UI, a GET request will return a list of configured servers as a JSON document. This document can be edited and sent as a POST to change the configuration.

In addition to the server, port, proto, cef, types, and filters keys, the v2 key is a Boolean which can be used to configure enhanced details for host and account syslog messages.

Appendix A contains examples of GET and POST requests. Described below, is the configuration of Syslog and Kafka filters that are not available in the UI.

In addition to the three toggle filters that are available in the UI, `triaged_detections`, `info_level_detection`, and `score_decreases`, six additional filters can be configured through only the API. The following table describes each filter.

FILTER	DESCRIPTION	NOTES
Triaged_detections	When false, do not include triaged detection syslog messages	Boolean
Info_level_detections	When false, do not include info level detection syslog messages	Boolean

FILTER	DESCRIPTION	NOTES
Score_decreases	When false, do not include host or account syslog messages when both threat/certainty decrease	Boolean
Host_threat_threshold	Set an integer threshold below which host syslog messages will not be sent	integer
Host_certainty_threshold	Set an integer threshold below which host syslog messages will not be sent	Integer
Host_observed_privilege_threshold	Set an integer threshold below which host syslog messages will not be sent	integer
Detection_categories	Detection syslog messages are only sent when included in this list, when the filter is enabled	list of strings
Detection_threat_threshold	Set an integer threshold below which detection syslog messages will not be sent	integer
Detection_certainty_threshold	Set an integer threshold below which detection syslog messages will not be sent	integer

A GET request can be used to obtain a list of currently-configured syslog or Kafka servers. The response will show a list of available filters. Filters can be configured by a POST request that includes the entire list of servers, as shown in Appendix A.

Every filter has an `enabled` property that is a Boolean indicating whether that filter is active. In addition, some filters require a `value` property that can be set to an integer value for the threshold level of the filter. Finally, detection_categories has a `values` property which is a list of strings to set which categories the filter applies to.

Review the listings in Appendix A for example of using GET and POST to configure Syslog and Kafka messaging.

IP Addresses

Version 2.5 of IP addresses API supports GET. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/ip_addresses`

The results will be returned as a list of ojects with the following fields in each object.

ELEMENT	DESCRIPTION	TYPE	NOTES
ip	The IP address.	integer	
first_seen	First time the IP address was seen.	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT
last_seen	Last time the IP address was seen.	string	Timestamp format: YYYY-MM-DD HH-MM-SS GMT

QUERY PARAMETER	DESCRIPTION
`include_ipv4`	Filters only IPV4 addresses
`include_ipv6`	Filters only IPV6 addresses

Detect Network Usage

Version 2.5 of the detect network usage API supports GET. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/usage/detect`

The following table lists the top-level fields and descriptions present in the response for a bulk GET request to the v2.5 API for detect usage.

ELEMENT	DESCRIPTION	TYPE	NOTES
`concurrent_ips`	The number of ip addresses on detect network separated by month	object	

There are no query parameters for the detect usage endpoint.

Traffic

Version 2.5 of traffic API supports GET. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/traffic`

The following table lists the top-level fields and descriptions present in the response for a bulk GET request to the v2.5 API for traffic stats.

ELEMENT	DESCRIPTION	TYPE	NOTES
count	The number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	Useful when using a web-based REST API browser.
previous	URL to the previous page of output	string	Useful when using a web-based REST API browser.
results	A list of all Traffic (described in the table below) being returned by the query	list	

The top-level response will be a list of traffic data.

There are no query parameters for this endpoint.

Vsensor

Version 2.5 of vsensor API supports GET. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/vsensor/*`

There are several different endpoints available:

- https://<vectra_management_ip>/api/v2.5/vsensor/vhd
- https://<vectra_management_ip>/api/v2.5/vsensor/qcow
- https://<vectra_management_ip>/api/v2.5/vsensor/info
- https://<vectra_management_ip>/api/v2.5/vsensor/qcow_info

There is no fields provided in the response object since this endpoint serves a file that will be downloaded.

There are no query parameters for this endpoint.

Audits

Audit information includes data that lists requested accesses to resources. This information includes but is not limited to:

- User
- Message describing action
- Result of action
- Timestamp

- Source IP

URL to retrieve audit information is `https://<vectra_management_ip>/api/v2.5/audits`.

An example of using curl to retrieve all hosts using token authentication:

```
curl –H “Authorization: Token db20f83b33744690e4168e7994c8dd0b53e64f94”
https://<vectra_management_ip>/api/v2.5/audits?start=2020-07-06&end=2020-07-07
```

The audits API using token authentication will return audits in the range {start, end}, inclusive. It expects the dates to be passed in the string form YYYY-MM-DD, interpreted as GMT. When start is not included in the request, it defaults to 0001-01-01. When end is not included, it defaults to 9999-12-31. Leaving out both start and end will return all audits stored on the host, which can be up to 200 MB.

The following table lists fields and description of the returned object.

ELEMENT	DESCRIPTION	TYPE	NOTES
start	The interpreted or assumed start date	string	YYYY-MM-DD, GMT
end	The interpreted or assumed end date	string	YYYY-MM-DD, GMT
audits	List of audits returned in the output	list of objects	Sorted such that audits[0] will be the oldest audit and audits[-1] will be the latest audit

The following table lists the fields and descriptions present in an audit. These audits will be contained inside the ‘audits’ field, which is a top-level field described in the table above.

ELEMENT	DESCRIPTION	TYPE	NOTES
user	Logged in user	string	
role	Role of user	string	Can be null (e.g. ssh’ing to Vectra Appliance)
version	The version of Vectra running on the Vectra Appliance	string	
dvchost	Hostname of Vectra Appliance accessed	string	If not configured, will be the Vectra Appliance’s IP
headend_addr	IP of Vectra Appliance accessed	string	
source_ip	IP of user accessing Vectra Appliance	string	
vectra_timestamp	Timestamp of event	string	The string contains an integer that represents the time in seconds since epoch

ELEMENT	DESCRIPTION	TYPE	NOTES
message	Describes what happened	string	
result	The status of the action	string	“success”, “pending”, or “failure”

The available options for Audits are listed below.

QUERY PARAMETER	DESCRIPTION
start	(Optional) Start date for audit, inclusive, formatted YYYY-MM-DD, in GMT
end	(Optional) End date for audit, inclusive, formatted YYYY-MM-DD, in GMT

Assignments

Assignments are used to assign host and account objects to analysts for investigation. This information includes but is not limited to:

- Assigned to user
- Assigned by user
- Date assigned
- Date resolved
- Events
- Outcome
- Host ID
- Account ID
- Triaged detections

URL to retrieve assignment information is `https://<vectra_management_ip>/api/v2.5/assignments`.

An example of using curl to retrieve all assignments using token authentication:

```
curl 'https://<vectra_management_ip>/api/v2.5/assignments/' \
-X 'GET' \
-H "Content-Type: application/json" \
--header "Authorization: Token 6296acad7107dd094a97aa555540f701cb264136" \
--compressed \
--insecure
```

Assignment Resolution

Once completed, assignments can be resolved. Assignment resolutions provide a way to label and track the outcomes of assignments. Outcomes get recorded in the assignment’s history.

Outcome choices may include one of the following built-in assignment outcomes:

- Benign True Positive
- Malicious True Positive
- False Positive

Alternatively, users can choose to define their own Custom outcomes for reporting purposes.

URL to resolve an assignment is
`https://<vectra_management_ip>/api/v2.5/assignments/<id>/resolve`.

An example of using curl to retrieve all assignments using token authentication:

```
curl 'https://<vectra_management_ip>/api/v2.5/assignments/9/resolve' \
-X 'GET' \
-H "Content-Type: application/json" \
--header "Authorization: Token 6296acad7107dd094a97aa555540f701cb264136" \
--data '{"outcome": 2, "note": "some note", "triage_as": "my triage rule",
"detection_ids": [23, 73, 85, 88]}' \
--compressed \
--insecure
```

The following table lists fields and description of the various elements for assignments.

ELEMENT	DESCRIPTION	TYPE	NOTES
count	Number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	
previous	URL link to the previous page of output	string	
results	List of users returned in the output	list of objects	

The following table lists the fields and descriptions present in an assignment. These elements will be contained inside the ‘results’ field, which is a top-level field described in the table above.

ELEMENT	DESCRIPTION	TYPE	NOTES
id	Object ID	integer	
assigned_by	The user that made the assignment	list of objects	
date_assigned	Timestamp from when the assignment was assigned	datetime	

ELEMENT	DESCRIPTION	TYPE	NOTES
date_resolved	Timestamp from when the assignment was resolved	datetime	
events	Actions take on the assignment	list of objects	
outcome	Outcome of the assignment	list of objects	
resolved_by	The user that resolved the assignment	list of objects	
triaged_detections	The number of detections that were triaged as a part of the resolution of this assignment	integer	
host_id	The ID of the host associated with this assignment	integer	
account_id	The ID of the account associated with this assignment	integer	
assigned_to	The user this assignment has been assigned to	list of objects	

The available options and filters for assignments are listed below.

QUERY PARAMETER	DESCRIPTION
accounts	Filter by accounts
hosts	Filter by hosts
assignees	Filter by assignees
resolution	Filter by resolution (outcome)
resolved	Filters by resolved status. True or False.
created_after	Filter by created after timestamp.

Examples of assignments queries

QUERY	COMMENT
`https://1.1.1.2/api/v2.5/assignments?accounts=1,2,3`	Retrieves all assignments on accounts 1, 2 and 3
`https://1.1.1.2/api/v2.5/assignments?hosts=1,2,3`	Retrieves all assignments on hosts 1, 2 and 3
`https://1.1.1.2/api/v2.5/assignments?assignees=1,2,3`	Retrieves all assignments for users 1, 2 and 3
`https://1.1.1.2/api/v2.5/assignments?resolution=1,2,3`	Retrieves all assignments with resolution 1, 2 or 3

QUERY	COMMENT
`https://1.1.1.2/api/v2.5/assignments?resolved=true`	Retrieves all assignments that have been resolved
`https://1.1.1.2/api/v2.5/assignments?created_after=2021-01-01T00:00:00Z`	Retrieves all assignments that have been created since 2021-01-01T00:00:00Z

Assignment Outcomes

Assignment outcomes are used to label resolution outcomes for assignments. This information includes but is not limited to:

- ID
- Title
- Category

URL to retrieve assignment outcome information is `https://<vectra_management_ip>/api/v2.5/assignment_outcomes`.

An example of using curl to retrieve all assignment outcomes using token authentication:

```
curl 'https://<vectra_management_ip>/api/v2.5/assignment_outcomes/' \
-X 'GET' \
-H "Content-Type: application/json" \
--header "Authorization: Token 6296acad7107dd094a97aa555540f701cb264136" \
--compressed \
--insecure
```

The following table lists fields and description of the various elements for assignment outcomes.

ELEMENT	DESCRIPTION	TYPE	NOTES
count	Number of object IDs retrieved in the output	integer	
next	URL to the next page of output	string	
previous	URL link to the previous page of output	string	
results	List of users returned in the output	list of objects	

The following table lists the fields and descriptions present in assignment outcomes. These elements will be contained inside the ‘results’ field, which is a top-level field described in the table above.

ELEMENT	DESCRIPTION	TYPE	NOTES
id	Object ID	integer	
builtin	True or false. Whether the outcome is a Vectra builtin outcome	Boolean	
user_selectable	True or false. Whether the outcome can be selected by GUI users.	Boolean	
title	The title of the outcome	string	
category	The category of the outcome. Categories include: • malicious_true_positive • benign_true_positive • false_positive	string	

Registration Token

A registration token can be used to automate sensor deployments in large scale environments. This information includes:

- Registration token
- Expiration date

URL to retrieve registration token information is `https://<vectra_management_ip>/api/v2.5/sensor_token`.

An example of using curl to retrieve the registration token using token authentication:

```
curl 'https://<vectra_management_ip>/api/v2.5/sensor_token/' \
-X 'GET' \
-H "Content-Type: application/json" \
--header "Authorization: Token 6296acad7107dd094a97aa555540f701cb264136" \
--compressed \
--insecure
```

The following table lists fields and description of the various elements for registration token.

ELEMENT	DESCRIPTION	TYPE	NOTES
token	Registration token	string	
expiration	Expiration date of registration token	string	

Examples of GET, POST and DELETE methods on the sensor_token endpoint are shown in Appendix A.

Settings

There are several different endpoints available stemming from the same starting endpoint of `https://<vectra_management_ip>/api/v2.5/settings/*`

There are several settings endpoints that all branch from the common base of /settings:

- https://<vectra_management_ip>/api/v2.5/settings/aws_connectors
- https://<vectra_management_ip>/api/v2.5/settings/internal_network
- https://<vectra_management_ip>/api/v2.5/settings/proxy
- https://<vectra_management_ip>/api/v2.5/settings/syslog_config
- https://<vectra_management_ip>/api/v2.5/settings/kafka_config

Each available endpoint will be described in more detail in the following sections.

HostID External Connectors for AWS

External connectors for AWS can be queried and added using the API, though the UI is required to enable AWS within the External Connectors settings page.

The URL to retrieve configured connectors is

https://<vectra_management_ip>/api/v2.5/settings/aws_connectors.

An example of using curl to retrieve the external connectors for AWS using token authentication:

```
curl 'https:// <vectra_management_ip>//api/v2.5/settings/aws_connectors' \
-X 'GET' \
-H "Content-Type: application/json" \
--header "Authorization: Token 6d72ec25a0d9a13143106e5810e0f084f0d4e459" \
--compressed \
--insecure
```

The following table lists fields and description of the various elements for external connectors for AWS.

ELEMENT	DESCRIPTION	TYPE	NOTES
access_key	AWS Access Key ID for the credentials	string	Required

ELEMENT	DESCRIPTION	TYPE	NOTES
account_type	The type of account being configured, either Single or Multiple	string	Required
alias	Descriptive name shown in the Vectra UI	string	Required
role_to_assume	Name of the IAM Role to assume witin AWS	string	
secret_key	AWS Secret Access Key for the credentials	string	

When adding or modifying an AWS Connector, the following body parameters are allowed inside the credentials list:

BODY PARAMETER	DESCRIPTION	REQUIRED
`acess_key`	Access key for AWS Account	Yes
`alias`	Alias for account in Vectra	Yes
`secret_key`	Secret key for AWS Account	Yes
`cloud_type`	One of 'AwsChinaGovernmentCloud', 'AwsPublicCloud', 'AwsUSGovernmentCloud'	Yes
`role_to_assume`	Role to assume in AWS	
`account_type`	Single or Multiple (multiple if child_accounts defined)	
`child_accounts`	List of objects: {'account' (string), 'role_name' (string)}	

Examples of GET and POST methods on the settings/aws_connectors endpoint are shown in Appendix A.

Internal Network

Top level information about subnet settings can be retrieved using the API at the endpoint:

https://<vectra_management_ip>/api/v2.5/settings/internal_network

The following table lists fields and description of the various elements for subnet settings.

ELEMENT	DESCRIPTION	TYPE	NOTES
included_subnets	Subnets to be included	list of strings	

ELEMENT	DESCRIPTION	TYPE	NOTES
excluded_subnets	Subnets to be excluded	list of strings	
dropped_subnets	Subnets that have been dropped	list of strings	

There are no query parameters for the /internal_network endpoint. When modifying the internal network settings, the following body parameters are allowed:

BODY PARAMETER	DESCRIPTION
`include`	Subnets to be included
`exclude`	Subnets to be excluded
`drop`	Subnets to drop

Proxy

Top level information about proxy settings can be retrieved using the API at the endpoint:

https://<vectra_management_ip>/api/v2.5/settings/proxy

The following table lists fields and description of the various elements for proxy settings.

ELEMENT	DESCRIPTION	TYPE	NOTES
host	Name of the proxied host	string	
enable	Whether proxy is enabled	bool	
port	Port of the proxied host	string	
authentication	Information on how the proxy is authenticated	object	Includes enabled status and associated username

Syslog Configuation

If any syslog destination has been setup, the corresponding syslog settings can be retrieved using the API at the endpoint:

https://<vectra_management_ip>/api/v2.5/settings/syslog_config

The following table lists fields and description of the various elements for syslog settings.

ELEMENT	DESCRIPTION	TYPE	NOTES
server	Address of destination syslog server	string	

ELEMENT	DESCRIPTION	TYPE	NOTES
port	Port of destination syslog server	string	
proto	Protocol used destination syslog server	string	Ex. tcp
cef	Event format	string	Ex. cef
types	Types of events to be sent via syslog	string	Ex. account, hosts…
filters	Filters used to prevent certain data from being transmitted	string	

Kafka Configuation

If any Kafka servers has been setup, the corresponding syslog settings can be retrieved using the API at the endpoint:

https://<vectra_management_ip>/api/v2.5/settings/kafka_config

The following table lists fields and description of the various elements for Kafka settings.

ELEMENT	DESCRIPTION	TYPE	NOTES
kafka_servers	Collection of information on Kafka servers	list of objects	Each object will have pertinent information to the corresponding Kafka server

Vectra Match Enablement

Version 2.5 of Vectra Match enablement API supports GET and POST. Changing the enablement state on a given device using the POST endpoint requires a license for Vectra Match, and an audit log will be emitted (NOTE: a valid Vectra Match license is not required to disable Match on a previously enabled device if your license has expired). The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/vectra-match/enablement`.

The enablement GET endpoint requires a single query parameters, device_serial:

QUERY PARAMETER	DESCRIPTION
device_serial	Serial number of the paired device in question

The following table lists the top-level fields present in the response for a GET request to the v2.5 API for Vectra Match enablement

ELEMENT	DESCRIPTION	TYPE	NOTES
is_enabled	Whether Vectra Match is enabled on the requested paired device	bool	
details	Error output	string	Only returned with an unsuccessful return code

The enablement POST endpoint requires a json body with two parameters, device_serial and desired_state (WARNING: POSTing to the enablement endpoint causes a reboot if changing state):

JSON BODY PARAMETER	DESCRIPTION
device_serial	Serial number of the paired device in question
desired_state	Boolean, true to enable, false to disable Vectra Match

The following table lists the top-level fields present in the response for a POST request to the v2.5 API for Vectra Match enablement

ELEMENT	DESCRIPTION	TYPE	NOTES
is_enabled	Whether Vectra Match is enabled on the requested paired device	bool	
details	Error output	string	Only present with an unsuccessful return code

Examples of GET and POST methods on the vectra-match/enablement endpoint are shown in Appendix A.

Vectra Match Stats

Version 2.5 of Vectra Match stats API supports GET. Use of the stats endpoint for Vectra Match does not require a valid license. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/vectra-match/stats`

The stats GET endpoint allows for a single optional query parameter, device_serial. If device_serial is not provided, information for all enabled and previously-enabled devices will be returned.

QUERY PARAMETER	DESCRIPTION
device_serial	Serial number of the paired device in question

The following table lists the top-level fields present in the response for a GET request to the v2.5 API for Vectra Match stats

ELEMENT	DESCRIPTION	TYPE	NOTES
stats	List of json objects, each of which represents stats from a paired device with Match enabled	list object	An example response can be found in Appendix A
details	Error output	string	Only present with an unsuccessful return code

Examples of the GET method on the vectra-match/stats endpoint is shown in Appendix A.

Vectra Match Status

Version 2.5 of Vectra Match status API supports GET. Use of the status endpoint for Vectra Match does not require a valid license. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/vectra-match/status`

The status GET endpoint allows for a single optional query parameter, device_serial. If device_serial is not provided, information for all enabled and previously-enabled devices will be returned.

QUERY PARAMETER	DESCRIPTION
device_serial	Serial number of the paired device in question

The following table lists the top-level fields present in the response for a GET request to the v2.5 API for Vectra Match status

ELEMENT	DESCRIPTION	TYPE	NOTES
status	List of json objects, each of which represents status of a paired device with Match enabled	list object	An example response can be found in Appendix A
details	Error output	string	Only present with an unsuccessful return code

Examples of the GET method on the vectra-match/status endpoint is shown in Appendix A.

Vectra Match Available Devices

Version 2.5 of Vectra Match available devices API supports GET. Use of the available devices endpoint for Vectra Match does not require a valid license. The available devices endpoint will return information about all currently paired sensors and, in the case of a mixed mode headend, information about the headend itself. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/vectra-match/available-devices`

The following table lists the top-level fields present in the response for a GET request to the v2.5 API for Vectra Match available devices

ELEMENT	DESCRIPTION	TYPE	NOTES
devices	List of json objects, each of which represents information about a paired device	list object	An example response can be found in Appendix A
details	Error output	string	Only present with an unsuccessful return code

Examples of the GET method on the vectra-match/available-devices endpoint is shown in Appendix A.

Vectra Match Rules

Version 2.5 of Vectra Match rules API supports GET, POST, and DELETE. Uploading a rules file requires a valid license for Vectra Match. Audit logs will be emitted for using the POST and DELETE endpoints. Once a rules file has been uploaded, the API will return a rules file uuid. This uuid will be used for managing the ruleset from that point forward, when interacting with the API. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/vectra-match/rules`.

The rules GET endpoint requires a single query parameter, uuid:

QUERY PARAMETER	DESCRIPTION
`uuid`	UUID of desired rules file

The following table lists the top-level fields present in the response for a GET request to the v2.5 API for Vectra Match rules

ELEMENT	DESCRIPTION	TYPE	NOTES
device_serials	List of all device serials on which the ruleset is currently running	list	If no assignments are present for a rules file, device_serials will be an empty list.
hashsum	sha256 sum of file	string	
name	Display name of file	string	Rules cannot be referenced in API operations by their display name, UUID must be used
notes	Notes about a file	object	Notes are added by the user when uploading a new rules file
timestamp	Timestamp of file upload time	string	ISO-8601 format
uuid	UUID of file	string	Use this UUID in later API calls to reference this ruleset
details	Error output	string	Only returned with an unsuccessful return code

The rules POST endpoint requires a multipart form upload with three fields. The ‘file’ field is required, while ‘notes’ and ‘rotate’ are optional. The ‘rotate’ field is only available in Vectra version 7.8+.

FORM FIELD	DESCRIPTION
file	File to upload
notes	Notes to store about the file (optional)
rotate	When this field is passed in equalling ‘true’, if the Vectra system is above the 25 file limit, the oldest unassigned file will be deleted automatically, and replaced by the file currently being uploaded, if possible. (optional)

Example of using POST endpoint with cURL:

```
curl --header "Authorization: Token <token>" --insecure -F file=@valid_rules.rules -F notes="This is a new file" -F rotate=true https://brain.ip/api/v2.5/vectra-match/rules
```

Example of using POST endpoint with Python Requests:

```
resp = requests.post("https://brain.dummy.local/api/v2.5/vectra-match/rules",
headers={"Authorization":"Token <token>"}, verify=False, files={"file":open("valid_rules.rules","rb")},
data={"notes":"This is a new file", “rotate”: “true”})
```

The following table lists the top-level fields present in the response for a POST request to the v2.5 API for Vectra Match rules

ELEMENT	DESCRIPTION	TYPE	NOTES
device_serials	List of all device serials on which the ruleset is currently running	list	If no assignments are present for a rules file, device_serials will be an empty list.
hashsum	sha256 sum of file	string	
name	Display name of file	string	Rules cannot be referenced in API operations by their display name, UUID must be used
notes	Notes about a file	object	Notes are added by the user when uploading a new rules file
timestamp	Timestamp of file upload time	string	ISO-8601 format
uuid	UUID of file	string	Use this UUID in later API calls to reference this ruleset
details	Error output	string	Only returned with an unsuccessful return code

The rules DELETE endpoint requires a json body with a single parameter, uuid:

JSON BODY PARAMETER	DESCRIPTION
`uuid`	UUID of desired rules file

The following table lists the top-level fields present in the response for a DELETE request to the v2.5 API for Vectra Match rules

ELEMENT	DESCRIPTION	TYPE	NOTES
details	Output	string	details will be 'success' if successful, otherwise, will contain information about the failure.

Examples of GET, POST, and DELETE methods on the vectra-match/rules endpoint are shown in Appendix A.

Vectra Match Assignment

Version 2.5 of Vectra Match assignment API supports GET, POST, and DELETE. Adding a new assignment via POST requires a valid license for Vectra Match. Audit logs will be emitted for using the POST and DELETE endpoints. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/vectra-match/assignment`.

The following table lists the top-level fields present in the response for a GET request to the v2.5 API for Vectra Match assignment

ELEMENT	DESCRIPTION	TYPE	NOTES
device_to_rules_map	List of objects, each of which represents a device, providing information about which ruleset is being run on that device, who updated it, and when it was last updated	list of objects	An example map can be found in Appendix A
rules_to_devices_map	List of objects, each of which represents a rules file, providing information about which devices that ruleset is being run on, and other metadata about the file	list of objects	An example map can be found in Appendix A
details	Error output	string	Only returned with an unsuccessful return code

The assignment POST endpoint requires a json body with two parameters, device_serials and uuid:

JSON BODY PARAMETER	DESCRIPTION
device_serials	List of serial numbers to assign the ruleset of <uuid> to
uuid	UUID of ruleset to assign to specified device(s)

The following table lists the top-level fields present in the response for a POST request to the v2.5 API for Vectra Match assignment

ELEMENT	DESCRIPTION	TYPE	NOTES
details	Output	string	details will be ‘success’ if successful, otherwise, will contain information about the failure.

The assignment DELETE endpoint requires a json body with two parameters, device_serial and uuid:

JSON BODY PARAMETER	DESCRIPTION
device_serial	Serial numbers to remove assignment from
uuid	UUID of ruleset

The following table lists the top-level fields present in the response for a DELETE request to the v2.5 API for Vectra Match assignment

ELEMENT	DESCRIPTION	TYPE	NOTES
details	Output	string	details will be ‘success’ if successful, otherwise, will contain information about the failure.

Examples of GET, POST, and DELETE methods on the vectra-match/assignment endpoint are shown in Appendix A.

Vectra Match Alert Stats

Version 2.5 of Vectra Match alert stats API supports GET. Use of the alert stats endpoint for Vectra Match does not require a valid license. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/vectra-match/alert-stats`

The alert stats GET endpoint allows for a single optional query parameter, device_serial. If device_serial is not provided, information for all enabled devices will be returned.

QUERY PARAMETER	DESCRIPTION
device_serial	Serial number of the paired device in question

The following table lists the top-level fields present in the response for a GET request to the v2.5 API for Vectra Match alert stats

ELEMENT	DESCRIPTION	TYPE	NOTES
alert_stats	List of json objects, each of which represents alert stats from a paired device with Match enabled	list object	An example response can be found in Appendix A
details	Error output	string	Only present with an unsuccessful return code

Examples of the GET method on the vectra-match/alert-stats endpoint is shown in Appendix A.

Vectra Match Download Vectra Ruleset

Version 2.5 of Vectra Match download Vectra ruleset API supports GET. Use of the download Vectra ruleset endpoint for Vectra Match does require a valid license. The API endpoint for accessing version 2.5 is `https://<vectra_management_ip>/api/v2.5/vectra-match/download-vectra-ruleset`

The download Vectra ruleset GET endpoint allows for no query parameters.

The Vectra Match download Vectra ruleset endpoint returns a stream of the contents of the curated.rules file.

Examples of the GET method on the vectra-match/download-vectra-ruleset endpoint is shown in Appendix A.

Health Events

The Health Events API provides access to health-related events generated by the Vectra platform and its sensors. Each event records the status of a health check, including details such as the entity type (device, brain, sensor), health check name, status code and label, output, and timestamps for the event and state changes. The API supports filtering, sorting, and pagination, allowing users to retrieve recent health events, monitor system health, and integrate this data into external dashboards or monitoring tools. Responses include a list of events, checkpoint information for pagination, and the count of remaining events

The URL to retrieve events is `https://<vectra_management_ip>/api/v2.5/events/health`

The following table lists the top-level fields and descriptions present in the response for a bulk GET request to the v2.5 API for Health Events.

ELEMENT	DESCRIPTION	TYPE	NOTES
events	A list of health events	list	Health Event fields are described in the table below
remaining_count	The number of remaining events	integer	
next_checkpoint	The next checkpoint value to use to retrieve any remaining events	integer	Example: "next_checkpoint": 101 Use with the ‘from’ query parameter described below

The following table lists the fields and descriptions present in an Health Event object.

Detailed examples of using GET is described in Appendix A

Element	Description	Type	Notes
event_object	Event type and name.	Object	The type of entity that caused the event to be produced. e.g. 'device' (For now we have no way to differentiate between brain or sensor.)
health_check_name	The name of the health check	string	

status_object	Status code and label	Object	Corresponds to syscheck return codes.
			OK = 0
			WARNING = 1
			CRITICAL = 2
			SKIP = 3
			UNKNOWN = 4
			TIMEOUT = 5
			INFO = 6
			ERROR = 11

output	Extra health check output data in JSON format. For sensu sources, this will be the output from a syscheck command in sensu format.	JSON	
event_timestamp	Time that the health check that triggered the event was executed at	string	Timestamp format: YYYY-MM-DD HH-MM-SS UTC
last_state_change_at	Time that the health check last changed state	string	Timestamp format: YYYY-MM-DD HH-MM-SS UTC
last_ok_at	Last time that it was OK	string	Timestamp format: YYYY-MM-DD HH-MM-SS UTC
last_state	The prior state of the health check	string	“OK”

The available query parameters to filter for Health Events are listed below.

QUERY PARAMETER	DESCRIPTION
entity_type	Filter by entity type (device, brain, sensor)
entity_name	Filter by entity name (case-insensitive contains)

QUERY PARAMETER	DESCRIPTION
health_check_name	Fliter by health check name (case-insensitive contains)
status	Filter by status code or label (e.g., 0, 1, OK, WARNING, etc.)
from	Used to provide a checkpoint value for filtering Events
limit	Used to specify the batch size returned by the response. By default 500 events will be returned if a limit is not specified. A maximum of 1,000 events will be returned per request.
event_timestamp_gte	Start date/time for scoring event, inclusive, formatted in ISO-8601
event_timestamp_lte	End date/time for scoring event, inclusive, formatted in ISO-8601
ordering	Orders events by “event_timestamp”. The default sorts event timestamps by ascending order. Events can be sorted in descending order by prepending the query with “minus” symbol.

Appendix A

This section will include examples of data retrieved from the REST API 2.5. Due to the amount of data that can be retrieved from a single query, the output examples below show only a snippet of the actual data that can be retrieved.

Note: The information in the following examples was generated in a lab environment. Any reference to IP addresses similar to those used in your environment is purely coincidental.

Triage Rules

GET

URL: `https://<vectra_management_ip>/api/v2.5/rules/<id>`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

```
{
    "id": 68,
    "url": "https://1.1.1.1/api/v2.5/rules/68",
    "description": "Expected behavior from these devices",
    "enabled": true,
    "created_timestamp": "2019-08-27T20:55:29Z",
    "last_timestamp": null,
    "is_whitelist": false,
    "priority": null,
    "active_detections": 2,
    "total_detections":3,
    "template": true,
    "additional_conditions": {
        "OR": [
            {
                "AND": [
                    {
                        "ANY_OF": {
                            "field": "remote1_port",
                            "values": [
                                {
                                    "url": null,
                                    "value": "135",
                                    "label": "135"
                                }
                            ],
                            "groups": [],
                            "label": "Port"
                        }
                    }
                ]
            }
        ]
    },
    "source_conditions": {
        "OR": [
            {
                "AND": [
                    {
                        "ANY_OF": {
                            "field": "host",
                            "values": [],
                            "groups": [
                                {
                                    "url": "https://192.168.51.36/api/v2.5/groups/8",
                                    "value": 8,
                                    "label": "Cognito - IPAM"
                                }
                            ],
                            "label": "Host"
                        }
                    }
                ]
            }
        ]
    },
    "detection_category": "RECONNAISSANCE",
    "triage_category": "Expected IPAM Behavior",
    "detection": "Internal Darknet Scan"
}
```

POST

This post method is used for creating a Triage Rule. There is also the support for using post to mark a detection(s) as custom.

URL: `https://<vectra_management_ip>/api/v2.5/rules`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:
```
{
      "description": "Peer to peer triage rule",
      "detection_category": "COMMAND & CONTROL",
      "triage_category": "Miscategorization",
      "detection": "Peer-to-Peer",
      "is_whitelist": false,
      "additional_conditions": {
        "OR": [
          {
            "AND": [
              {
                "ANY_OF": {
                  "field": "remote1_ip",
                  "values": [],
                  "groups": [
                    {
                      "value": 2
                    }
                  ]
                }
              },
              {
                "ANY_OF": {
                  "field": "remote1_dns",
                  "values": [
                    {
                      "value": "test.server.com"
                    }
                  ],
                  "groups": [
                    {
                      "value": 11
                    }
                  ]
                }
              }
            ]
          }
        ]
      },
      "source_conditions": {
        "OR": [
          {
            "AND": [
              {
                "ANY_OF": {
                  "field": "host",
                  "values": [
                    {
                      "value": 1
                    }
```

```
                  ],
                  "groups": [
                    {
                      "value": 8
                    }
                  ]
                }
              }
            ]
          }
        ]
      }
    }
```

The following fields are mandatory:

- “detection_category”: Must be set to the category of the detection – LATERAL MOVEMENT, RECONNAISSANCE, COMMAND & CONTROL, EXFILTRATION, BOTNET, INFO
- “detection”: The detection that must be triaged. Use the detection name as seen in the UI or the “Understanding Vectra Detections” guide
- “triage_category”: The name that will be used for the triaged detection. Only applies if “is_whitelist” is set to 0
- “description”: User defined description for the rule
- “is_whitelist”: A Boolean flag indicating whether to create a “whitelist” or a “track without scores” rule
- source_conditions: conditions that can be applied to the source of a detection. Can be NULL
- additional_conditions: other conditions that are different on a per-detection-type basis. Can be NULL

Response:

Success
The response contains the id of the triage rule if successful:

```
{
    "_meta": {
        "message": "Successfully created triage filter",
        "level": "success"
    },
    "id": 11
}
```

Failure
Failure will result in an error message along with an indication of what was incorrect

```
{
    "_meta": {
        "message": "Invalid field(s) found",
        "level": "error"
    },
    "host": [
        "Invalid host: 5 does not map to a real host"
    ]
}
```

PUT

PUT method can be used to modify existing triage filters. A PUT is a full override, so the format will be the same as POST. In order to use PUT to add additional IPs or Hosts to existing triage filters, first use GET to get the existing triage filter for that id, extend the host or IP list with new data and then do a PUT using the complete list. Just performing a PUT with the new list without including existing IPs or hosts will override the existing configuration.

URL: `https://<vectra_management_ip>/api/v2.5/rules/<id>`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:
```
{
    "detection_category": "LATERAL MOVEMENT",
    "triage_category": "Susp Rmt Exec - Test",
    "detection": "Suspicious Remote Execution",
    "is_whitelist": 0,
    "description": "put test",
        "additional_conditions": {
          "OR": [
            {
              "AND": [
                {
                  "ANY_OF": {
                    "field": "remote1_ip",
                    "values": [
                      {
                        "value": "1.1.1.1"
                      }
                    ],
                    "groups": [],
                  }
                }
              ]
            }
          ]
        },
        "source_conditions": {
          "OR": [
            {
              "AND": [
                {
                  "ANY_OF": {
                    "field": "ip",
                    "values": [
                      {"value": "1.1.1.1"},
                      {"value": "1.2.1.1"},
                      {"value": "1.1.3.1"}
                    ],
                    "groups": []
                  }
                }
              ]
            }
          ]
        }

}
```

DELETE

This endpoint is used for deleting existing Triage rules.

URL: `https://<vectra_management_ip>/api/v2.5/rules/<id>`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:
```
{
       "detectionIdLIst": [<detection_id1>, <detection_id2>, …]
}
```

Detections

GET

URL: `https://<vectra_management_ip>/api/v2.5/detections`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
{
  "last_timestamp": "2019-08-27T20:13:08Z",
  "grouped_details": [
    {
```

```
    "num_sessions": 10,
    "protocol": "tcp",
    "last_timestamp": "2019-08-27T20:13:08Z",
    "host_detection": 36,
    "accounts": [],
    "is_host_detail": true,
    "bytes_received": 3425075,
    "dst_geo": null,
    "src_ip": "1.1.1.1",
    "dst_ips": [
      "35.166.75.118"
    ],
    "grouping_field": "last_timestamp",
    "description": null,
    "is_account_detail": false,
    "dst_ports": [
      8080
    ],
    "account_detection": null,
    "first_timestamp": "2019-08-27T20:13:08Z",
    "dst_geo_lat": null,
    "dst_geo_lon": null,
    "bytes_sent": 141982,
    "target_domains": [
      "www.linkedin.com:443"
    ],
    "account_uid": null
  }
],
"custom_detection": null,
"is_custom_model": false,
"detection": "Hidden HTTP Tunnel",
"detection_type": "Hidden HTTP Tunnel",
"is_targeting_key_asset": false,
"note_modified_timestamp": null,
"c_score": 59,
"t_score": 10,
"id": 36,
"category": "COMMAND & CONTROL",
"src_ip": "1.1.1.1",
"detection_category": "COMMAND & CONTROL",
"note": "This is a second note by Admin.",
"note_modified_by": "admin",
"note_modified_timestamp": "2021-12-15T17:26:41Z",
"notes": [
  {
      "id": 15,
      "date_created": "2021-12-15T17:26:41Z",
        "date_modified": null,
        "created_by": "admin",
        "modified_by": "admin",
        "note": "This is a second note by Admin"
    },
    {
        "id": 14,
        "date_created": "2021-12-15T17:26:13Z",
        "date_modified": null,
        "created_by": "admin",
        "modified_by": "admin",
        "note": "This is a note."
    }
  ],
  "state": "active",
  "sensor": "CfxAb1HK",
  "assigned_date": null,
  "targets_key_asset": false,
  "description": null,
  "tags": [],
  "triage_rule_id": null,
  "first_timestamp": "2019-08-27T20:13:08Z",
  "campaign_summaries": [],
  "groups": [],
  "detection_url": "https://172.168.51.16/api/v2.5/detections/36",
  "src_account": null,
  "sensor_name": "Vectra X",
  "url": "https:// 172.168.51.16/api/v2.5/detections/36",
  "certainty": 59,
  "is_marked_custom": false,
  "note_modified_by": null,
  "summary": {
    "num_sessions": 10,
    "dst_ips": [
      "35.166.75.118"
    ],
    "bytes_sent": 141982,
    "dst_ports": [
      8080
    ],
    "bytes_received": 3425075
  },
  "threat": 10,
  "assigned_to": null,
  "is_triaged": false,
  "src_host": {
    "is_key_asset": false,
    "threat": 90,
    "name": "IP-1.1.1.1",
    "groups": [],
    "url": "https://192.168.51.36/api/v2.5/hosts/2",
    "ip": "1.1.1.1",
    "certainty": 99,
    "id": 2
  }
}
```

PATCH to add a note to a Detection

URL: `https://<vectra_management_ip>/api/v2.5/detections/<id>`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
 “Content-Type”: “application/json”
```

Body:

```
{
    "note": “This is a detection”,
}
```

PATCH to Mark/Unmark detections as Fixed in bulk

URL: `https://<vectra_management_ip>/api/v2.5/detections/`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
"Content-Type": "application/json"
```

Body:

```
{
    "detectionIdList": [<det_id1>, <det_id2>, …],
    "mark_as_fixed": "True/False"
}
```

Detections (Notes)

GET

URL: `https://<vectra_management_ip>/api/v2.5/detections/<detection_id>/notes`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

```
[
  {
    "id": 1,
    "date_created": "2021-01-11T13:42:43Z",
    "date_modified": null,
    "created_by": "vadmin",
    "modified_by": null,
    "note": "this is a detection note"
  },
  {
    "id": 2,
    … … …
  }
]
```

POST
URL: https://<vectra_management_ip>/api/v2.5/detections/<detection_id>/notes

Headers:
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
"Content-Type": "application/json"

Body:
{"note":"this is a detection note"}

Response:

```
{
  "id": 2,
  "date_created": "2021-01-11T14:14:10.527603Z",
  "date_modified": null,
  "created_by": "vadmin",
  "modified_by": null,
  "note": "this is a detection note"
}
```

GET
URL:
https://<vectra_management_ip>/api/v2.5/detections/<detection_id>/notes/<detection_id>
Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```
Response:
```
{
  "id": 1,
  "date_created": "2021-01-11T13:54:47.987918Z",
  "date_modified": null,
  "created_by": "vadmin",
  "modified_by": null,
  "note": " this is a detection note "
}
```

PATCH

URL:

`https://<vectra_management_ip>/api/v2.5/detections/<detection_id>/notes/<detection_id>`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:
```
{"note":"updated note"}
```

Response:
```
{
  "id": 1,
  "date_created": "2021-01-11T13:47:42Z",
  "date_modified": "2021-01-11T13:57:11Z",
  "created_by": "vadmin",
  "modified_by": "vadmin",
  "note": "updated note"
}
```

DELETE

URL:

`https://<vectra_management_ip>/api/v2.5/detections/<detection_id>/notes/<detection_id>`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Close a single Detection

PATCH

URL: `https://<vectra_management_ip>/api/v2.5/detections/<detection_id>/close`

Headers:
```
"Authorization": "Token <api-key>"
"Content-Type": "application/json"
```

Body:

```
{
  "reason": “remediated”
}
```

Response:

```
{
    "_meta": {
        "level": "Success",
        "message": " Successfully closed detection as remediated"
    }
}
```

Close multiple Detections

PATCH

URL: `https://<vectra_management_ip>/api/v2.5/detections/close`

Headers:

```
"Authorization": "Token <api-key>"
"Content-Type": "application/json"
```

Body:

```
{
  "detectionIdList": [123, 456],
  "reason": “remediated”
}
```

Response:

```
{
    "_meta": {
        "level": "Success",
        "message": " Successfully closed detections as remediated"
    }
}
```

Open a single Detection

PATCH

URL: `https://<vectra_management_ip>/api/v2.5/detections/<detection_id>/open`

Headers:

```
"Authorization": "Token <api-key>"
"Content-Type": "application/json"
```

Response:
```
{
    "_meta": {
        "level": "Success",
        "message": " Successfully re-opened detection"
    }
}
```

Open multiple Detections

PATCH

URL: `https://<vectra_management_ip>/api/v2.5/detections/open`

Headers:
```
"Authorization": "Token <api-key>"
"Content-Type": "application/json"
```

Body:
```
{
  "detectionIdList": [123, 456],
}
```

Response:
```
{
    "_meta": {
        "level": "Success",
        "message": " Successfully re-opened detections"
    }
}
```

Account

GET

URL: `https://<vectra_management_ip>/api/v2.5/accounts`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
{
  "id": 18,
  "url": "https://192.168.51.36/api/v2.5/accounts/18",
  "name": "mixitupab.anon.anoncorp.com",
  "state": "inactive",
  "threat": 0,
  "certainty": 0,
"severity": "Low",
"account_type": ["kerberos"],
"tags": [],
"note": "This is an important note by Admin",
"note_modified_by": "admin",
"note_modified_timestamp": "2021-12-15T16:49:21Z",
"notes": [
  {
      "id": 12,
      "date_created": "2021-12-15T16:49:21Z",
      "date_modified": null,
      "created_by": "admin",
      "modified_by": "admin",
      "note": "This is an important note by Admin"
  },
  {
      "id": 11,
      "date_created": "2021-12-15T16:48:51Z",
      "date_modified": null,
      "created_by": "admin",
      "modified_by": "admin",
      "note": "This is another from Admin"
  }
],
"privilege_level": null,
"privilege_category": null,
"last_detection_timestamp": "2019-08-28T19:05:12Z",
"detection_set": [
  "https://192.168.51.36/api/v2.5/detections/64"
],
"detection_summaries": [
  {
    "tags": [],
    "detection_type": "Privilege Anomaly: Unusual Host",
    "is_targeting_key_asset": false,
    "detection_id": 64,
    "detection_url": "https://192.168.51.36/api/v2.5/detections/64",
    "certainty": 0,
    "detection_category": "LATERAL MOVEMENT",
    "summary": {
      "src_accounts": [
        {
          "name": "mixitupab.anon.anoncorp.com",
          "privilege_category": null,
          "privilege_level": null,
          "id": 18
        }
      ],
        "src_hosts": [
          {
            "name": "IP-1.1.1.1",
            "privilege_category": null,
            "privilege_level": null,
            "id": 2
          }
        ],
        "services_accessed": [
          {
            "name": "http/calm.mine.way.grand.com",
            "privilege_category": null,
            "privilege_level": null,
            "id": null
          },
          {
            "name": "http/reply.kiss.art.prompt.com",
            "privilege_category": null,
            "privilege_level": null,
            "id": null
          },
        ]
      },
      "state": "active",
      "threat": 0,
      "assigned_to": null,
      "assigned_date": null,
      "is_triaged": false
    }
  ],
  "ldap": null
}
```

Account (Notes)

GET

URL: `https://<vectra_management_ip>/api/v2.5/accounts/<account_id>/notes`

Headers:

`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

```
[

  {

    "id": 1,

    "date_created": "2021-01-11T13:42:43Z",
    "date_modified": null,
    "created_by": "vadmin",
    "modified_by": null,
    "note": "this is an account note"
  },
  {
    "id": 2,
    … … …
  }
]
```

POST
URL: `https://<vectra_management_ip>/api/v2.5/accounts/<account_id>/notes`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:
```
{"note":"this is a note"}
```

Response:

```
{
  "id": 2,
  "date_created": "2021-01-11T14:14:10.527603Z",
  "date_modified": null,
  "created_by": "vadmin",
  "modified_by": null,
  "note": "this is a note"
}
```

GET
URL: `https://<vectra_management_ip>/api/v2.5/accounts/<account_id>/notes/<note_id>`
Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```
Response:
```
{
  "id": 1,
  "date_created": "2021-01-11T13:54:47.987918Z",
  "date_modified": null,
  "created_by": "vadmin",
  "modified_by": null,
  "note": " this is an account note "
}
```

PATCH
URL: `https://<vectra_management_ip>/api/v2.5/accounts/<account_id>/notes/<note_id>`
Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:
```
{"note":"updated note"}
```

Response:
```
{
  "id": 1,
  "date_created": "2021-01-11T13:47:42Z",
  "date_modified": "2021-01-11T13:57:11Z",
  "created_by": "vadmin",
  "modified_by": "vadmin",
  "note": "updated note"
}
```

DELETE
URL: `https://<vectra_management_ip>/api/v2.5/accounts/<account_id>/notes/<note_id>`
Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Close an Account

PATCH
URL: `https://<vectra_management_ip>/api/v2.5/accounts/<account_id>/close`

Headers:
```
"Authorization": "Token <api-key>"
"Content-Type": "application/json"
```

Body:
```
{
  "reason": “remediated”
}
```

Response:

```
{
    "_meta": {
        "level": "Success",
        "message": " Successfully closed account as remediated"
    },
    "detections_closed": [123, 456]
}
```

Host

GET

URL: https://<vectra_management_ip>/api/v2.5/hosts

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

```
{
    "id": 1029,
    "name": "insightws07",
    "active_traffic": true,
    "t_score": 90,
    "c_score": 99,
    "last_source": "10.16.6.6",
    "previous_ips": [
        "10.16.12.1",
        "10.16.0.1",
     ],
    "last_detection_timestamp": "2019-08-28T19:05:12Z",
    "key_asset": false,
    "state": "active",
    "targets_key_asset": true,
    "probable_owner": "dkelle",
    "detection_set": [
        "https://10.1.6.10/api/v2.5/detections/1354",
        "https://10.1.6.10/api/v2.5/detections/1365",
        "https://10.1.6.10/api/v2.5/detections/1380",
        "https://10.1.6.10/api/v2.5/detections/1388",
        "https://10.1.6.10/api/v2.5/detections/1410",
        "https://10.1.6.10/api/v2.5/detections/1435",
        "https://10.1.6.10/api/v2.5/detections/1436",
        "https://10.1.6.10/api/v2.5/detections/1476"
    ],
    "host_artifact_set": [
        {
            "type": "kerberos",
            "value": "insightws07"
        }
    ],
    "sensor": null,
    "tags": [],
    "note": "This is an important note by Admin",
    "note_modified_by": "admin",
    "note_modified_timestamp": "2021-12-15T16:49:21Z",
    "notes": [
      {
          "id": 12,
          "date_created": "2021-12-15T16:49:21Z",
          "date_modified": null,
          "created_by": "admin",
          "modified_by": "admin",
          "note": "This is an important note by Admin"
      },
      {
          "id": 11,
          "date_created": "2021-12-15T16:48:51Z",
          "date_modified": null,
          "created_by": "admin",
          "modified_by": "admin",
          "note": "This is another from Admin"
      }
    ],
    "url": "https://10.1.6.10/api/v2.5/hosts/1029"
}
```

PATCH to set or remove the key asset flag on a host

URL: `https://<vectra_management_ip>/api/v2.5/hosts/<id>`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:
```
{
    "key_asset": "True"
}
```

Host (Notes)

GET
URL: `https://<vectra_management_ip>/api/v2.5/hosts/<host_id>/notes`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

```
[
  {
    "id": 1,
    "date_created": "2021-01-11T13:48:31Z",
    "date_modified": null,
    "created_by": "vadmin",
    "modified_by": null,
    "note": "create this note"
  },
  {
    "id": 2,
    … … …
  },
]
```

POST

URL: `https://<vectra_management_ip>/api/v2.5/hosts/<host_id>/notes`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:

```
{"note":"this is a note"}
```

Response:

```
{
  "id": 2,
  "date_created": "2021-01-11T13:54:47.987918Z",
  "date_modified": null,
  "created_by": "vadmin",
  "modified_by": null,
  "note": "this is a note"
}
```

GET

URL: `https://<vectra_management_ip>/api/v2.5/hosts/<host_id>/notes/<note_id>`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

```
{
  "id": 1,
  "date_created": "2021-01-11T13:54:47.987918Z",
  "date_modified": null,
  "created_by": "vadmin",
  "modified_by": null,
  "note": "another note"
}
```

PATCH

URL: `https://<vectra_management_ip>/api/v2.5/hosts/<host_id>/notes/<note_id>`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:

```
{"note":"updated note"}
```

Response:

```
{
  "id": 1,
  "date_created": "2021-01-11T13:47:42Z",
  "date_modified": "2021-01-11T13:57:11Z",
  "created_by": "vadmin",
  "modified_by": "vadmin",
  "note": "updated note"
}
```

DELETE

URL: `https://<vectra_management_ip>/api/v2.5/hosts/<host_id>/notes/<note_id>`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Close a Host

PATCH

URL: `https://<vectra_management_ip>/api/v2.5/hosts/<host_id>/close`

Headers:

```
"Authorization": "Token <api-key>"
"Content-Type": "application/json"
```

Body:

```
{
  "reason": “remediated”
}
```

Response:

```
{
    "_meta": {
        "level": "Success",
        "message": " Successfully closed host as remediated"
    },
    "detections_closed": [123, 456]
}
```

Users

GET

URL: `https://<vectra_management_ip>/api/v2.5/users/`

Headers:

`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

```
{
    "count": 3,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "last_login": "2019-08-27T20:55:29Z ",
            "username": "admin",
            "email": "ad@vectra.ai",
            "account_type": "local",
            "authentication_profile": null,
            "role": "Super Admin"
        },
        {
            "id": 2,
            "last_login": "2019-08-27T20:55:29Z ",
            "username": "vadmin",
            "email": "vadmin@vectra.ai",
            "account_type": "TACACS",
            "authentication_profile": tacacs-profile,
            "role": "Admin"
        },
        {
            "id": 3,
            "last_login": "2019-08-27T20:55:29Z ",
            "username": "cognito",
            "email": "cognito@vectra.ai",
            "account_type": "LDAP",
            "authentication_profile": ldap-profile,
            "role": "Security Analyst"
        }
    ]
}
```

PATCH to change a user to TACACS

URL: `https://<vectra_management_ip>/api/v2.5/users/<id>`

```
“Authorization”: “Token <api-key>”
“Content-Type”: “application/json”
```

Body:

```
{
    "account_type": "TACACS",
    "authentication_profile": "tacacs-profile",
}
```

threatFeeds

GET

URL: `https://<vectra_management_ip>/api/v2.5/threatFeeds/<id>/`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

```
{
    "name": "test",
    "data": null,
    "_rev": "1-69a246ab1d2e970732ea34e195a47486",
    "uploadResults": null,
    "lastUpdatedBy": "admin",
    "lastUpdated": "2017-11-07T18:37:02.517125+00:00",
    "version": "3.11-240-g3ccd27c",
    "_id": "4de02b1bad3f4ff028847c30ef74d5f7",
    "type": "STIX",
    "defaults": {
        "duration": 7,
        "category": "cnc",
        "certainty": "High",
        "indicatorType": "Anonymization"
    },
    "uploadDate": "2017-11-07T18:37:02.517125+00:00"
}
```

POST to create a threatFeed

URL: `https://<vectra_management_ip>/api/v2.5/threatFeeds/`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Body:

```
{
    "threatFeed": {
           "name":"test-feed",
            "defaults": {
                "certainty":"High",
                "duration":7,
                "indicatorType":"Anonymization",
                "category":"cnc"
             }
      }
}
```

All the fields shown above are mandatory. See the section on threatFeeds for possible values each label can take.

POST to add or replace STIX file to an existing threatFeed

URL: `https://<vectra_management_ip>/api/v2.5/threatFeeds/<id>/`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Body: Send the STIX file as a multi part form data

```
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="file"; filename="stix_test_url_anon.xml"
Content-Type: text/xml


------WebKitFormBoundary7MA4YWxkTrZu0gW—
```

Response Success:

```
{
    "threatFeed": {
        "uploadResults": {
            "category": "cnc",
            "certainty": "High",
            "expiration": "2019-08-27T20:55:29.27Z ",
            "indicatorType": "Anonymization",
            "observableCount": 4
        }
    }
}
```

If you are using a tool like postman, perform the POST with the body of the POST as “form-data” with Key as “file” and value as the selected file from the file system. Do not add a content type header explicitly for postman, since the tool does it automatically.

Proxy

GET

URL: `https://<vectra_management_ip>/api/v2.5/settings/proxy`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response if a proxy is configured:
```
{
    "proxy": {
            "host": "proxy.com",
            "enable": true,
            "port": "8888",
            "authentication": {
                    "enable": false,
                    "user": ""
            }
    }
}
```

Response if a proxy is not configured:
```
{
    "proxy": {
            "host": "",
            "enable": false,
            "port": "",
            "authentication": {
                    "enable": false,
                    "user": ""
            }
    }
}
```

Proxies

GET

URL: `https://<vectra_management_ip>/api/v2.5/proxies/`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
{
    "meta": {
        "count": 3
    },
    "proxies": [
        {
            "source": "user",
            "id": "f62afac59c994b9c9c18d04b1e1095c4",
            "considerProxy": true,
            "address": "1.2.3.5"
        },
        {
            "source": "user",
            "id": "57c9042358a64cbaaf13dc1a2e0b43b9",
            "considerProxy": true,
            "address": "10.1.2.5"
        },
        {
            "source": "user",
            "id": "1955087a53cb4f3aa825a405a7024b34",
            "considerProxy": true,
            "address": "1.2.3.7"
        }
    ]
}
```

POST to create a proxy

URL: `https://<vectra_management_ip>/api/v2.5/proxies/`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```
“Content-Type”: “application/json”

Body:
```
{
  "proxy": {
    "address": "1.2.3.8",
    "considerProxy": true
  }
}
```

All the fields shown above are mandatory.

PATCH to modify a proxy object

URL: https://<vectra_management_ip>/api/v2.5/proxies/<id>/

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:
```
{
  "proxy": {
    "address": "1.2.3.9",
    "considerProxy": true
  }
}
```

Patch operation can only be used for user defined proxies, not for proxy objects that have the source as “Cognito”.

Tagging

GET to retrieve tags for a host

URL: https://<vectra_management_ip>/api/v2.5/tagging/host/<host_id>

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
{
  "status": "success",
  "tag_id": "1000",
  "tags": [
    "test",
    "this is a tag",
    "We need to follow up on this host."
  ]
}
```

PATCH to add “new_tag” for a host

URL: https://<vectra_management_ip>/api/v2.5/tagging/host/<host_id>

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:

```
{
  "tags": [“test”, “new_tag”, “this is a tag”, “We need to follow up on this host.”]
}
```

PATCH to clear tags for a detection

URL: `https://<vectra_management_ip>/api/v2.5/tagging/detection/<detection_id>`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:

```
{

  "tags": []

}
```

Patch operation will alter the tags for the host or detection to match the “tags” list provided in the PATCH.

Groups

GET (all)

URL: `https://<vectra_management_ip>/api/v2.5/groups`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

```
[
	{
		"id": 1,
		"name": "Test Group",
		"description": "desc",
		"last_modified": "2022-11-08T16:28:03Z",
		"last_modified_by": "API Client d8ce639e",
		"type": "account",
		"members": [],
		"rules": [],
		"regex": null,
		 "membership_evaluation_ongoing": false,
		 "member_count": 0,
        “built_using”: “static_members”,
        “ad_group_dn”: null

    },
    {
        "id": 2,
        "name": "New Group",
        "description": "desc 2",
        "last_modified": "2022-11-17T15:13:25Z",
        "last_modified_by": "API Client d8ce639e",
        "type": "account",
        "members": [
            {
                "uid": “AWS:400102379209”
            },
            {
                "uid": “name_2”
            }
        ],
        "rules": [],
        "regex": "[\\-\\.\\w\\d]*\\d ",
        "membership_evaluation_ongoing": false,
        "member_count": 2,
        “built_using”: “regex”,
        “ad_group_dn”: null

    }
]
```

GET (single)

URL: `https://<vectra_management_ip>/api/v2.5/groups/1`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

<u>Success</u>
The response contains the id of the group if successful:

```
{
    "id": 1,
    "name": "Test Group",
    "description": "desc",
    "last_modified": "2022-11-08T16:28:03Z",
        "last_modified_by": "API Client d8ce639e",
        "type": "account",
        "members": [],
        "rules": [],
        "regex": null,
        "membership_evaluation_ongoing": false,
        "member_count": 0,
        “built_using”: “static_members”,
        “ad_group_dn”: null
}
```

Failure

If the id is not valid, the response will indicate that the group could not be found.

```
{
        "detail": "Not found."
}
```

POST

URL: `https://<vectra_management_ip>/api/v2.5/groups/<id>`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Static Group Body:

```
{
        “name”: “New Group Name”,
        “description”: “New description text”,
        “type”: “account”,
        “members”: [“uid_1”, “uid_2”]
}
```

Regex Group Body:

```
{
        “name”: “New Regex Group Name”,
        “description”: “New description text”,
        “type”: “host”,
        "regex": "IP-1[\\w\\d\\.]*"
}
```

AD Group Body:

```
{
        “name”: “New AD Group Name”,
        “description”: “New description text”,
        “type”: “host”,
        "ad_group_dn": "CN=Administrators,CN=test,DC=test,DC=test"
}
```

The following fields are mandatory:

- "name": group name is required and must be unique
- "type": group type is required and is one of "account, domain, host, ip".
- "description": group description is required
- "members": zero or more members are allowed
- "regex" : must provide valid regex if creating dynamic group
- "ad_group_dn": must provide valid distinguished name if creating AD group

Response:

Success
The response contains the id of the group if successful:

```
{
        "group": {
                "id": 52
        }
}
```

Failure
Failure will result in an error message along with an indication of what was incorrect. Shown below is the error when trying to create a group without specifying a name.

```
{
    "_meta": {
        "message": "Invalid field(s) found",
        "level": "error"
    },
    "name": [
        "This field may not be null."
    ]
}
```

The group names must be unique, so specifying a duplicate name will result in the following 409 response.

```
{
        "_meta": {
                "level": "error",
                "message": "Attempt to create group with duplicate name: Duplicate Group"
        }
}
```

PATCH

PATCH method can be used to modify existing groups. A PATCH is a partial override, so the format can include one or more fields from the POST.

When using the default PATCH functionality to modify the members, a complete new list must be included. For example, if a group already contains hosts 1, 2, 3, 4, and the group should be modified to include host 5, then 5 should be added to the list, the members field of the PATCH request should be 1, 2, 3, 4, 5.

Alternatively, the “membership_action” query parameter can be passed in with a value of “replace”, “append”, or “remove”. The “append” and “remove” values will only affect members listed in the request body; if a user wants to add or remove host 1 for a group, then the user can pass the members field of 1 in the request body with the appropriate value of “membership_action”. The “replace” value will perform the default PATCH behavior.

Regex groups may be patched but their members cannot be manually changed. You can transition regex groups to static groups, and static groups to regex groups, and AD groups to static groups.

URL: `https://<vectra_management_ip>/api/v2.5/groups/<id>`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json”
```

Body:
```
{
       “name”: “new test rename”,
       “description”: “desc”,
       “type”: “account”,
       “members”: [],
}
```

Response:

Success
The response contains the id of the group if successful:

```
{
       "id": 4,
       "name": "new test rename",
       "description": "desc",
       "last_modified": "2022-11-17T14:58:23Z",
       "last_modified_by": "API Client d8ce639e",
       "type": "account",
       "members": [],
       "rules": [],
       "regex": "",
       "membership_evaluation_ongoing": false,
       "member_count": 0,
       "built_using": "static_members",
        “ad_group_dn”: null
}
```

Failure

Failure will result in an error message along with an indication of what was incorrect. Shown below is the error when trying to edit a group when the id is not valid. The response will indicate that the group could not be found.

```
{
        "detail": "Not found."
}
```

Static groups can be transitioned to dynamic groups when regex is defined.

Transition Static Group to Dynamic Group Body:

```
{
        “name”: “New Group Name”,
        “description”: “New description text”,
        “type”: “account”,
        “members”: [],
        “regex”: "[\\d\\w\\.\\-]*@vectra.ai"
}
```

Response:

Success

The response contains the id of the group if successful:

```
{
        "id": 4,
        "name": "New Group Name",
        "description": "New description text",
        "last_modified": "2022-11-17T14:58:23Z",
        "last_modified_by": "API Client d8ce639e",
        "type": "account",
        "members": [],
        "rules": [],
        "regex": "[\\d\\w\\.\\-]*@vectra.ai",
        "membership_evaluation_ongoing": true,
        "member_count": 0,
        "built_using": "regex",
        “ad_group_dn”: null
}
```

Failure

Failure will result in an error message along with an indication of what was incorrect. Shown below is the error when trying to transition to a dynamic group while specifying members.

```
{
    "_meta": {
        "level": "error",
        "message": "Members cannot be specified when creating a regex group. Please
only provide a regex. Members will be added upon creation."
    }
}
```

Dynamic groups can be transitiond to static groups when regex is defined as null.

Transition Dynamic Group to Static Group Body:

```
{
       “name”: “New Group Name”,
       “description”: “New description text”,
       “type”: “account”,
       “members”: [],
       “regex”: null
}
```

Response:

Success
The response contains the id of the group if successful:

```
{
       "id": 4,
       "name": "New Group Name",
       "description": "New description text",
       "last_modified": "2022-11-17T14:58:23Z",
       "last_modified_by": "API Client d8ce639e",
       "type": "account",
       "members": [],
       "rules": [],
       "regex": “”,
       "membership_evaluation_ongoing": false,
       "member_count": 0,
       "built_using": "static_members"
}
```

Failure
Failure will result in an error message along with an indication of what was incorrect. Shown below is the error when trying to edit a group when the id is not valid. The response will indicate that the group could not be found.

```
{
       "detail": "Not found."
}
```

AD groups can be transitiond to static groups when the distinguished name is defined as null.

Transition AD Group to Static Group Body:

```
{
       “name”: “New Group Name”,
       “description”: “New description text”,
       “ad_group_dn”: null
}
```

Response:

<u>Success</u>
The response contains the id of the group if successful:

```
{
       "id": 4,
       "name": "New Group Name",
       "description": "New description text",
       "last_modified": "2022-11-17T14:58:23Z",
       "last_modified_by": "API Client d8ce639e",
       "type": "account",
       "members": [],
       "rules": [],
       "regex": null,
       "membership_evaluation_ongoing": false,
       "member_count": 0,
       "built_using": "static_members",
       “ad_group_dn”: null
}
```

<u>Failure</u>
Failure will result in an error message along with an indication of what was incorrect. Shown below is the error when trying to transition an AD group to a static group while specifying regex.

```
{
    "_meta": {
        "level": "error",
        "message": "Invalid parameters, unable to edit regex for AD Group"
    }
}
```

DELETE
URL: `https://<vectra_management_ip>/api/v2.5/groups/<id>`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Group Members

GET
```
URL: https://<vectra_management_ip>/api/v2.5/groups/<group_id>/members
```

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response - Host:
```
{
    "count": 2100,
    "next": "https://<vectra_management_ip>/api/v2.5/groups/89/members?page=3",
    "previous": " https://<vectra_management_ip>/api/v2.5/groups/89/members?page=2",
    "results": [
        {
             "id": “19026”,
             "name": “IP-1.1.1.1”,
             "is_key_asset": “false”,
             "url": “https://<vectra_management_ip>/api/v2.5/hosts/19026”,
        },
        {
            "id": “19078”,
             "name": “IP-1.4.3.2”,
             "is_key_asset": “true”,
             "url": “https://<vectra_management_ip>/api/v2.5/hosts/19078”,
        }
    ],
}
```

```
Response - Account:
{
    "count": 321,
    "next": "https://<vectra_management_ip>/api/v2.5/groups/90/members?page=3",
    "previous": " https://<vectra_management_ip>/api/v2.5/groups/90/members?page=2",
    "results": [
        {
             "uid": “a99@vectra.ai”,
        },
        {
            "uid": “O365:a99@vectra.ai”,
        }
    ],
}
```

```
Response - IP:
{
    "count": 341,
    "next": "https://<vectra_management_ip>/api/v2.5/groups/91/members?page=3",
    "previous": " https://<vectra_management_ip>/api/v2.5/groups/91/members?page=2",
    "results": [
      “3.7.35.0/25”,
      “3.21.137.128/25”,
      “3.25.41.128/25”,
      “13.52.6.128/25”,
      “16.63.29.0/24”…
    ],
}


Response - Domain:
{
    "count": 341,
    "next": "https://<vectra_management_ip>/api/v2.5/groups/92/members?page=3",
    "previous": " https://<vectra_management_ip>/api/v2.5/groups/92/members?page=2",
    "results": [
      “*.slack.com”,
      “*.slack-files.com”,
      “*.slack-msgs.com”,
      “*.slack-imgs.com”
    ],
}
```

Active Directory Groups

GET

URL: `https://<vectra_portal_url>/api/v2.5/settings/active_directory/groups`

Headers:

`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

```
{
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
             "ad_profile_name": “test_ad”,
             "dn": “CN=Administrators,CN=test,DC=test_ad,DC=test”,
             "existing_host_group": false,
             "existing_account_group": true
        },
        {
             "ad_profile_name": “test_ad”,
             "dn": “CN=Desktop,CN=test_ad,DC=test”,
            "existing_host_group": false,
            "existing_account_group": false
        }
    ]
}
```

Health

GET

URL: `https://<vectra_management_ip>/api/v2.5/health`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:
```
{
    "network": {
        "interfaces": {
            "brain": {
                "eth0": {
                    "speed_mbps": 1000,
                    "duplex": "FULL",
                    "link": "UP"
                },
                "eth1": {
                    "link": "DOWN"
                }
            },
            "sensors": {}
        },
        "traffic": {
            "brain": {
                "interface_peak_traffic": {
                    "eth3": {
                        "peak_traffic_mbps": 0
                    },
                    "eth2": {
                        "peak_traffic_mbps": 0
                    },
                    "eth1": {
                        "peak_traffic_mbps": 0
                    },
                    "eth0": {
                        "peak_traffic_mbps": 0
                    }
                },
                "aggregated_peak_traffic_mbps": 0
            },
            "sensors": {}
        },
        "vlan_count": 0
    },
    "system": {
        "serial_number": "S11181714785673",
        "uptime": "35 days, 1 hours, 5 minutes",
        "version": {
            "last_update": "Fri Nov 22 20:07:40 2019",
            "mode": "mixed",
            "vectra_version": "5.3.0-0-4742",
            "model": "X4"
        }
    },
    "memory": {
        "free_bytes": 47405064192,
        "dimm_status": [
            {
                "status": "OK",
                "dimm": "mc1"
            },
            {
                "status": "OK",
                "dimm": "mc0"
            }
        ],
        "used_bytes": 87754235904,
        "usage_percent": 39.5,
        "total_bytes": 135159300096
    },
    "sensors": [
        {
            "status": "paired",
            "name": "Test Sensor APP Team",
            "serial_number": "V422ry78u5q5673s8koq4orxc8fb96ac6",
            "package_version": "4.9.0-13-31",
            "last_seen": "2019-11-26T16:16:39Z",
            "ip_address": "192.168.5.238",
            "luid": "mnbtl29o",
            "location": "None"
        }
    ],
    "disk": {
        "raid_disks_missing": {
            "status": "OK",
            "output": {},
            "error": ""
        },
        "disk_utilization": {
            "free_bytes": 34955042816,
            "total_bytes": 62861103104,
            "usage_percent": 44.39,
            "used_bytes": 27906060288
        },
        "degraded_raid_volume": {
            "status": "OK",
            "output": [],
            "error": ""
        },
        "disk_raid": {
            "status": "OK",
            "output": "",
            "error": ""
        }
    },
    "cpu": {
        "idle_percent": 66.8,
        "user_percent": 32.5,
        "nice_percent": 0,
        "system_percent": 0.5
    },
    "hostid": {
        {
            "ip_always": 0,
            "ip_sometimes": 0,
            "ip_never": 0,
            "artifact_counts": {
                "proxy_ip": 0,
                "cookie": 0,
                "kerberos": 0,
                "clear_state": 0,
                "dns": 0,
                "crowdstrike": 0,
                "idle_end": 0,
                "src_port": 0,
                "split": 0,
                "vmachine_info": 0,
                "kerberos_user": 0,
                "arsenic": 0,
                "end_time": 0,
                "mdns": 0,
                "idle_start": 0,
                "uagent": 0,
                "static_ip": 0,
                "carbon_black": 0,
                "dhcp": 0,
                "netbios": 0,
                "total": 0,
                "rdns": 0
            }
        }
    },
    "power": {
        "status": "OK",
        "power_supplies": {
            "1": {
                "faults": [],
                "temperature_celcius": 31.0
            }
        },
        "error": ""
    }
}
```

GET

URL: `https://<vectra_management_ip>/api/v2.5/health/connectivity`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
{
    "connectivity": {
        "sensors": [
            {
                "name": "E1976E3611050022",
                "output": {
                    "last_check": "2021-10-12T22"
                },
                "error": "",
                "status": "OK",
                "serial_number": "E1976E3611050022",
                "luid": "snvv4hm0",
                "ip_address": "192.168.1.1"
            },
            {
                "name": "other sensor",
                "output": {
                    "affected_metadata_hours": [
                        "2021-10-12-21",
                        "2021-10-12-22"
                    ],
                    "last_check": "2021-10-12T22"
                },
                "error": "exception trying to connect to remote database for past 2
hours",
                "status": "CRITICAL",
                "serial_number": "E1976E3611050023",
                "luid": "l3vs98lu",
                "ip_address": "192.168.1.2"
            }
        ]
    }
}
```

GET

URL: `https://<vectra_management_ip>/api/v2.5/health/trafficdrop`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

```
{
  "trafficdrop": {
    "sensors": [
      {
        "name": "192.168.54.217",
        "output": [
          {
            "end": "2021-11-30T16:00:00+00:00",
            "interfaces": [
              {
                "baseline": 5625000,
                "cutoff": 2250000,
                "name": "eth0",
                "traffic": 0
              }
            ],
            "start": "2021-11-30T08:00:00+00:00"
          }
        ],
        "error": "At least one interface had low traffic volume",
        "status": "WARNING",
        "serial_number": "V422cae26ebb40b5a923b3ec0087cacb6",
        "luid": "mcq3ggiq",
        "ip_address": "192.168.54.217"
      }
    ]
  }
}
```

Account Lockdown

GET

URL: `https://<vectra_management_ip>/api/v2.5/lockdown/account`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
{
    "lock_date": "2020-02-21T15:59:24Z",
    "locked_by": "admin",
    "unlock_date": "2020-02-21T16:59:24Z",
    "account_id": 79,
    "account_name": "Lockdown_User_1@redwoods.test"
    “type”: “host”,
}
```

Host Lockdown

GET

URL: `https://<vectra_management_ip>/api/v2.5/lockdown/host`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
[
  {
    "host_id": 10,
    "locked_by": "vadmin",
    "unlock_date": "2020-06-09T15:21:06Z",
    "host_name": "def-1",
    "lock_date": "2020-06-09T14:21:06Z"
  }
]
```

Syslog/Kafka

GET

URL: https://<vectra_management_ip>/api/v2.5/settings/kafka_config

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
{
    "syslog_servers": [
        {
            "cef": "cef",
            "proto": "udp",
            "server": "<server_ip>",
            "filters": {
                "info_level_detections": {
                    "enabled": false
                },
                "detection_categories": {
                    "values": [],
                    "enabled": false
                },
                "detection_certainty_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "detection_threat_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "triaged_detections": {
                    "enabled": false
                },
                "host_certainty_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "score_decreases": {
                    "enabled": false
                },
                "host_threat_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "host_observed_privilege_threshold": {
                    "enabled": false,
                    "value": 0
                }
            },
            "port": 514,
            "types": [
                "host",
                "detection"
            ],
            "v2": false,
        },
        {},
        {}
    ]
}
```

GET

URL: `https://<vectra_management_ip>/api/v2.5/settings/kafka_config`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

```
{
    "kafka_servers": [
        {
            "cef": "cef",
            "proto": "udp",
            "bootstrap_servers": "<server_ip>:<port>",
            "filters": {
                "info_level_detections": {
                    "enabled": false
                },
                "detection_categories": {
                    "values": [],
                    "enabled": false
                },
                "detection_certainty_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "detection_threat_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "triaged_detections": {
                    "enabled": false
                },
                "host_certainty_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "score_decreases": {
                    "enabled": false
                },
                "host_threat_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "host_observed_privilege_threshold": {
                    "enabled": false,
                    "value": 0
                }
            },
            "port": 514,
            "types": [
                "host",
                "detection"
            ],
            "v2": false,
        },
        {},
        {}
    ]
}
```

POST

URL: `https://<vectra_management_ip>/api/v2.5/settings/syslog_config`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"

BODY/json:
{
    "syslog_servers": [
        {
            "cef": "cef",
            "proto": "udp",
            "server": "<server_ip>",
            "filters": {
                "info_level_detections": {
                    "enabled": false
                },
                "detection_categories": {
                    "values": [],
                    "enabled": false
                },
                "detection_certainty_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "detection_threat_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "triaged_detections": {
                    "enabled": false
                },
                "host_certainty_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "score_decreases": {
                    "enabled": false
                },
                "host_threat_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "host_observed_privilege_threshold": {
                    "enabled": false,
                    "value": 0
                }
            },
            "port": 514,
            "types": [
                "host",
                "detection"
            ]
            "v2": true,
        },
        {},
        {}
    ]
}
```

POST

URL: `https://<vectra_management_ip>/api/v2.5/settings/kafka_config`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

```
BODY/json:
{
    "syslog_servers": [
        {
            "cef": "cef",
            "proto": "udp",
            "bootstrap_servers": "<server_ip>:<port>",
            "filters": {
                "info_level_detections": {
                    "enabled": false
                },
                "detection_categories": {
                    "values": [],
                    "enabled": false
                },
                "detection_certainty_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "detection_threat_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "triaged_detections": {
                    "enabled": false
                },
                "host_certainty_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "score_decreases": {
                    "enabled": false
                },
                "host_threat_threshold": {
                    "enabled": false,
                    "value": 0
                },
                "host_observed_privilege_threshold": {
                    "enabled": false,
                    "value": 0
                }
            },
            "types": [
                "host",
                "detection"
            ],
            "v2": true,
        },
        {},
        {}
    ]
}
```

Assignments

GET

Used to retrieve assignments

URL: `https://<vectra_management_ip>/api/v2.5/assignments`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json"
```

Response:

```
{
    "count": 3,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 4,
            "assigned_by": {
                "id": 1,
                "username": "josem"
            },
            "date_assigned": "2021-05-18T04:34:48Z",
            "date_resolved": null,
            "events": [
                {
                    "assignment_id": 4,
                    "actor": 1,
                    "event_type": "created",
                    "datetime": "2021-05-18T04:34:48Z",
                    "context": {
                        "to": 20
                    }
                }
            ],
            "outcome": null,
            "resolved_by": null,
            "triaged_detections": {},
            "host_id": 1,
            "account_id": null,
            "assigned_to": {
                "id": 20,
                "username": "admin"
            }
        },
        {
            "id": 2,
            "assigned_by": {
                "id": 1,
                "username": "josem"
            },
            "date_assigned": "2021-05-18T03:51:38Z",
            "date_resolved": "2021-05-18T03:51:53Z",
            "events": [
                {
                    "assignment_id": 2,
                    "actor": 1,
                    "event_type": "resolved",
                    "datetime": "2021-05-18T03:51:53Z",
                    "context": {
                        "triage_as": "Miscategorization",
                        "detection_ids": [
                            1
                        ],
                        "created_rule_ids": [
                            27
                        ]
                    }
                },
                {
                    "assignment_id": 2,
                    "actor": 1,
                    "event_type": "created",
                    "datetime": "2021-05-18T03:51:38Z",
                    "context": {
                        "to": 20
                    }
                }
            ],
            "outcome": {
                "id": 3,
                "builtin": true,
                "user_selectable": true,
                "title": "False Positive",
                "category": "false_positive"
            },
            "resolved_by": {
                "id": 1,
                "username": "josem"
            },
            "triaged_detections": [
                1
            ],
            "host_id": 1,
            "account_id": null,
            "assigned_to": {
                "id": 20,
                "username": "admin"
            }
        },
        {
            "id": 1,
            "assigned_by": {
                "id": 1,
                "username": "josem"
            },
            "date_assigned": "2021-05-17T23:21:52Z",
            "date_resolved": "2021-05-17T23:23:53Z",
            "events": [
                {
```

```
                    "assignment_id": 1,
                    "actor": 1,
                    "event_type": "resolved",
                    "datetime": "2021-05-17T23:23:53Z",
                    "context": {
                        "triage_as": null,
                        "detection_ids": [],
                        "created_rule_ids": null
                    }
                },
                {
                    "assignment_id": 1,
                    "actor": 1,
                    "event_type": "created",
                    "datetime": "2021-05-17T23:21:52Z",
                    "context": {
                        "to": 20
                    }
                }
            ],
            "outcome": {
                "id": 1,
                "builtin": true,
                "user_selectable": true,
                "title": "Benign True Positive",
                "category": "benign_true_positive"
            },
            "resolved_by": {
                "id": 1,
                "username": "josem"
            },
            "triaged_detections": {},
            "host_id": 1,
            "account_id": null,
            "assigned_to": {
                "id": 20,
                "username": "admin"
            }
        }
    ]
}
```

POST

Used to assign an entity to a user.

URL: `https://<vectra_management_ip>/api/v2.5/assignments`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json"
```

```
BODY/json:
{
    "assign_host_id": "1",
    "assign_to_user_id": "1"
}
```

PUT

Used to modify/reassign assignments.

URL: `https://<vectra_management_ip>/api/v2.5/assignments/<id>`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json"
```

```
BODY/json:
{
    "assign_to_user_id": "1"
}
```

DELETE

Used to delete assignments.

URL: `https://<vectra_management_ip>/api/v2.5/assignments/<id>`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json"
```

PUT

Used to resolve assignments.

URL: `https://<vectra_management_ip>/api/v2.5/assignments/<id>/resolve`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json"

BODY/json:
{
    "outcome": "1",
    "note": "Resolved by JoseM",
    "triage_as": "My triage rule",
    "detection_ids": [1,2,3]
}
```

Assignment Outcomes

GET

Used to retrieve assignment outcomes

URL: `https://<vectra_management_ip>/api/v2.5/assignment_outcomes`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json"
```

Response:

```
{
    "count": 4,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "builtin": true,
            "user_selectable": true,


            "title": "Benign True Positive",
            "category": "benign_true_positive"
        },
        {
            "id": 2,
            "builtin": true,
            "user_selectable": true,
            "title": "Malicious True Positive",
            "category": "malicious_true_positive"
        },
        {
            "id": 3,
            "builtin": true,
            "user_selectable": true,
            "title": "False Positive",
            "category": "false_positive"
        },
        {
            "id": 6,
            "builtin": false,
            "user_selectable": true,
            "title": "Custom Outcome",
            "category": "benign_true_positive"
        }
    ]
}
```

POST

Used to create assignment outcomes

URL: `https://<vectra_management_ip>/api/v2.5/assignment_outcomes`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json"
```

```
BODY/json:
{
    "title": "Custom outcome",
    "category": "benign_true_positive"
}
```

PUT

Used to modify an assignment outcome. Title can always be modified. Category can only be modified if assignment outcome has not been used as an outcome for assignment.

URL: `https://<vectra_management_ip>/api/v2.5/assignment_outcomes/<id>`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json"
```

```
BODY/json:
{
    "title": "My New Outcome Title",
    "category": "false_positive"
}
```

DELETE

Used to delete an assignment outcome.

URL: `https://<vectra_management_ip>/api/v2.5/assignment_outcomes/<id>`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
“Content-Type”: “application/json"
```

Registration Token

GET

Used to retrieve sensor registration token.

URL: `https://<vectra_management_ip>/api/v2.5/sensor_token`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
{
    "token": "rqblghnqkcggbtwphhldndjvrbggkzsz",
    "expiration": "2022-03-02T19:10:21Z"
}
```

POST

Used to create sensor registration token.

URL: `https://<vectra_management_ip>/api/v2.5/sensor_token`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

```
{
    "token": "grchqrryltlwfjpsgcbktynlrvtnxndb",
    "expiration": "2022-03-02T19:26:38.674Z"
}
```

DELETE

Used to delete a sensor registration token.

URL: `https://<vectra_management_ip>/api/v2.5/sensor_token`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:
204 No Content

Settings - HostID AWS External Connectors

GET

Used to retrieve AWS external connectors.

URL: `https://<vectra_management_ip>/api/v2.5/settings/aws_connectors`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:
```
{
  "credentials": [
    {
      "access_key": "1234123412341234",
      "alias": "saas-production-account",
      "cloud_type": AwsPublicCloud,
      "secret_key": "***********",
      "status": {
        "credValid": false,
        "message": "Credentials provided are invalid"
      },
      "role_to_assume": "VectraCognitoHostID",
      "account_type": "Single"
    }
  ]
}
```

POST

Used to create and remove AWS external connectors.

URL: `https://<vectra_management_ip>/api/v2.5/settings/aws_connectors`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
Status code: 200
{}
```

```
Example Python script to retrieve connectors.

import json
import requests
vectra_url = 'https://192.168.49.232/api/v2.5/settings/aws_connectors'
headers = {'Content-Type': 'application/json', 'Authorization': 'Token
6d72ec23a0d9a13143106e1810e0f084f0d4e455'}
request_body = {
    "credentials": [
        {
            "access_key": "1234123412341234",
            "alias": "example-alias",
            "secret_key": "1234123412341234",
            "role_to_assume": "VectraCognitoHostID",
            "account_type": "Single",
        }
    ]
}
response = requests.post(url=vectra_url, data=json.dumps(request_body), verify=False,
headers=headers)
print(response.json())

# Multi Account Type example
import json
import requests
vectra_url = 'https://192.168.49.232/api/v2.5/settings/aws_connectors'
headers = {'Content-Type': 'application/json', 'Authorization': 'Token
6d72ec23a0d9a13143106e1810e0f084f0d4e455'}
request_body = {
    "credentials": [
        {
            "access_key": "1234123412341234",
            "alias": "example-alias",
            "secret_key": "1234123412341234",
            "role_to_assume": "VectraCognitoHostID",
            "account_type": "Multiple",
            "child_accounts": [
                {
                    “account”: “123456789”,
                    “role_name”: “example_role”
                 }
             ],
        }
    ]
}
response = requests.post(url=vectra_url, data=json.dumps(request_body), verify=False,
headers=headers)
print(response.json())
```

To remove a connector, remove it from the request payload.

```
import json
import requests
vectra_url = 'https://192.168.49.232/api/v2.5/settings/aws_connectors'
headers = {'Content-Type': 'application/json', 'Authorization': 'Token
6d72ec23a0d9a13143106e1810e0f084f0d4e455'}
request_body = {
    "credentials": []
}
response = requests.post(url=vectra_url, data=json.dumps(request_body), verify=False,
headers=headers)
print(response.json())
```

Settings – Internal Network

GET

Used to retrieve internal network data.

URL: `https://<vectra_management_ip>/api/v2.5/settings/internal_network`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:
```
{
    "included_subnets": [
        "10.0.0.0/8",
        "172.16.0.0/12",
        "192.168.0.0/16",
        "35.164.89.51/32"
    ],
    "excluded_subnets": [],
    "dropped_subnets": []
}
```

POST

Used to set internal network settings.

URL: `https://<vectra_management_ip>/api/v2.5/settings/internal_network`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Body:
```
{
    “exclude”: [“10.0.0.0/24”]
}
```

Response:
```
{
    "included": [],
    "excluded": [
        {
            "net_id": "10.0.0.0",
            "mask": 24
        }
    ],
    "dropped": []
}
```

Settings – Proxy

GET

Used to retrieve internal proxy data.

URL: `https://<vectra_management_ip>/api/v2.5/settings/proxy`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
```
{
    "proxy": {
        "host": "",
        "enable": false,
        "port": "",
        "authentication": {
            "enable": false,
            "username": ""
        }
    }
}
```

Settings – Syslog Configuration

GET

Used to retrieve syslog configuration information.

URL: `https://<vectra_management_ip>/api/v2.5/settings/syslog_config`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

```
{
    "syslog_servers": [
        {
            "server": "sccorpceflog.vectra.io",
            "port": "514",
            "proto": "tcp",
            "cef": "cef",
            "types": [
                "account_detection",
                "lockdown",
                "account",
                "audit",
                "campaigns",
                "detection",
                "host_lockdown",
                "host",
                "health",
                "vectra_match"
            ],
            "filters": {
                "triaged_detections": {
                    "enabled": true,
                    "info": {
                        "type": "boolean",
                        "syslog_field_name": "triaged",
                        "syslog_message_types": [
                            "detection"
                        ],
                        "message_when_true": "Include filtered detections",
                        "message_when_false": "Do not include filtered detections"
                    }
                },
                "info_level_detections": {
                    "enabled": true,
                    "info": {
                        "type": "boolean",
                        "syslog_field_name": "category",
                        "syslog_message_types": [
                            "detection"
                        ],
                        "message_when_true": "Include detections in info category",
                        "message_when_false": "Do not include detections in info
category"
                    }
                },
                "score_decreases": {
                    "enabled": false,
                    "info": {
                        "type": "boolean",
                        "syslog_field_name": "score_decreases",
                        "syslog_message_types": [
                            "host"
                        ],
                        "message_when_true": "Send syslog when score decreases",
                        "message_when_false": "Do not send syslog when score
decreases"
                    }
                },
                "host_threat_threshold": {
                    "enabled": false,
                    "value": -1,
                    "info": {
                        "type": "threshold",
                        "syslog_field_name": "threat",
                        "syslog_message_types": [
                            "host"
                        ]
                    }
                },
                "host_certainty_threshold": {
                    "enabled": false,
                    "value": -1,
                    "info": {
                        "type": "threshold",
                        "syslog_field_name": "certainty",
                        "syslog_message_types": [
                            "host"
                        ]
                    }
                },
                "host_observed_privilege_threshold": {
                    "enabled": false,
                    "value": -1,
                    "info": {
                        "type": "threshold",
                        "syslog_field_name": "privilege",
                        "syslog_message_types": [
                            "host"
                        ]
                    }
                },
                "detection_categories": {
                    "enabled": false,
                    "values": [],
                    "info": {
                        "type": "detection_match",
                        "syslog_field_name": "category",
                        "syslog_message_types": [
                            "detection"
                        ]
                    }
                },
                "detection_threat_threshold": {
                    "enabled": false,
                    "value": -1,
                    "info": {
                        "type": "threshold",
                        "syslog_field_name": "threat",
                        "syslog_message_types": [
                            "detection"
                        ]
                    }
                },
                "detection_certainty_threshold": {
                    "enabled": false,
                    "value": -1,
                    "info": {
                        "type": "threshold",
                        "syslog_field_name": "certainty",
                        "syslog_message_types": [
                            "detection"
                        ]
                    }
```

```
                }
            },
            "v2": false,
            "filters_to_display": [
                "triaged_detections",
                "info_level_detections",
                "score_decreases"
            ]
        },
        {},
        {}
    ]
}
```

Settings – Kafka Configuration

GET

Used to retrieve syslog configuration information.

URL: `https://<vectra_management_ip>/api/v2.5/settings/syslog_config`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

```
{
    "kafka_servers": [
        {},
    ]
}
```

Campaigns

GET

Used to retrieve Campaigns.

URL: `https://<vectra_management_ip>/api/v2.5/campaigns`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 14633,
            "dst_ip": "44.234.86.110",
            "target_domain": "cs-brain.vectracloudlab.com",
            "state": "active",
            "name": "cs-brain.vectracloudlab.com-25",
            "last_updated": "2023-05-14T17:10:41Z",
            "note": null,
            "note_modified_by": null,
            "note_modified_timestamp": null
        }
     ]
}
```

IP Addresses

GET

Used to retrieve IP address information.

URL: `https://<vectra_management_ip>/api/v2.5/ip_addresses`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:

```
[
    {
        "ip": "3.3.3.3",
        "first_seen": "2023-05-31T21:34:52.589603Z",
        "last_seen": "2023-05-31T21:34:52.589603Z"
    }
]
```

Detect Usage

GET

Used to retrieve information on detect network usage.

URL: `https://<vectra_management_ip>/api/v2.5/usage/detect`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

```
{
    "concurrent_ips": {
        "months": {
            "2023-05": 2393
        }
    }
}
```

Subnets

GET

Used to retrieve Subnets.

URL: `https://<vectra_management_ip>/api/v2.5/subnets`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

```
{
    "count": 1,
    "next": "https://<vectra_management_ip>//api/v2.5/subnets",
    "previous": null,
    "results": [
        {
            "subnet": "10.96.96.0",
            "hosts": 1,
            "lastSeen": "2023-05-17 15:11:13.084055-07:00",
            "firstSeen": "2023-05-17 15:11:13.084055-07:00"
        },
     ]
}
```

Traffic

GET

Used to retrieve traffic stats.

URL: `https://<vectra_management_ip>/api/v2.5/traffic`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

```
{
    "count": 1,
    "next": "https://<vectra_management_ip>/api/v2.5/traffic",
    "previous": null,
    "results": [
        {
            "time": "2023-05-17T14:15:00Z",
            "dhcp_pkt": 148,
            "dhcp_pct": 1.344077525810466e-06,
            "dns_pkt": 123568,
            "dns_pct": 0.0011221957547928897,
            "duplicate_pkt": 54478840,
            "duplicate_pct": 0.4947553005150287,
            "etv_other_pkt": 75128643,
            "etv_other_pct": 0.6822886527090392,
            "http_pkt": 1478980,
            "http_pct": 0.013431512021102453,
            "https_pkt": 29960217,
            "https_pct": 0.27208685363584234,
            "in-to-in_pkt": 42302391,
            "in-to-in_pct": 0.3841736015618036,
            "in-to-out_pkt": 3523056,
            "in-to-out_pct": 0.0319950026471062,
            "kerberos_pkt": 1553,
            "kerberos_pct": 1.4103732416105768e-05,
            "mdns_pkt": 85,
            "mdns_pct": 7.719364168506055e-07,
            "netbios_pkt": 80,
            "netbios_pct": 7.265283923299816e-07,
            "other_pkt": 78547126,
            "other_pct": 0.7133339646865062,
            "out-to-in_pkt": 2814377,
            "out-to-in_pct": 0.02555905996525596,
            "out-to-out_pkt": 6310289,
            "out-to-out_pct": 0.05730755152884459,
            "packet_count": 110112696,
            "smb_pkt": 938,
            "smb_pct": 8.518545400069034e-06,
            "smtp_pkt": 1,
            "smtp_pct": 9.08160490412477e-09,
            "ssl_pkt": 33378700,
            "ssl_pct": 0.30313216561330947,
            "interfaces": [
                {
                    "bandwidth": 2833,
                    "name": "all"
                }
            ],
            "timestamp": "05/17/2023, 07:15",
            "bandwidth": 2833
        }
    ]
}
```

Vectra Match Enablement

GET

Used to retrieve whether Vectra Match is enabled on a given device

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/enablement`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

```
Query Params:
{“device_serial”:”<device_serial>”}
```

Response:
Status Code: 200
```
{
  "is_enabled": true
}
```

POST

Used to enable or disable Vectra Match on a given device (WARNING: POSTing to the enablement endpoint causes a reboot if changing state). A valid Vectra Match license is required to enable this feature on a given paired device, but is not required to disable Match.

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/enablement`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

```
Required Request JSON Body:
{“device_serial”:”<device_serial”, “desired_state”: true}
```

Response:
```
Status code: 200
```

```
{
  "is_enabled": true
}
```

Vectra Match Stats

GET

Used to retrieve Vectra Match stats. Returned fields include:

- eve_stats: statistics directly pulled from the Vectra Match (suricata) engine
- interface_stats: statistics about each network interface and its processing/drop rate
- ft_stats: statistics about each flow thread and its processing/drop rate
- forwarder_stats: statistics about how many syslog messages Vectra Match is producing
- device_serial: serial of the device for which the stats are given
- timestamp: timestamp of when the information was gathered

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/stats`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"

Optional Query Params:
{“device_serial”:”<device_serial>”}
```

Response:
Status Code: 200
```
{
  "stats": [
    {
      "eve_stats": {
        "timestamp": "2023-02-01T01:05:45.123456+00:00",
        "event_type": "stats",
        "stats": {
          "uptime": 177,
          "capture": {
            "packets": 1361,
            "rx_errors": 0,
            "tx_errors": 0,
            "dpdk": {
              "imissed": 0,
              "no_mbufs": 0,
              "ierrors": 0
            }
          },
          "decoder": {
            "pkts": 1361,
            "bytes": 2335792,
"invalid": 0,
"ipv4": 0,
"ipv6": 0,
"ethernet": 1361,
"chdlc": 0,
"raw": 0,
"null": 0,
"sll": 0,
"tcp": 0,
"udp": 0,
"sctp": 0,
"esp": 0,
"icmpv4": 0,
"icmpv6": 0,
"ppp": 0,
"pppoe": 0,
"geneve": 0,
"gre": 0,
"vlan": 0,
"vlan_qinq": 0,
"vxlan": 0,
"vntag": 0,
"ieee8021ah": 0,
"teredo": 0,
"ipv4_in_ipv6": 0,
"ipv6_in_ipv6": 0,
"mpls": 0,
"avg_pkt_size": 1716,
"max_pkt_size": 2048,
"max_mac_addrs_src": 0,
"max_mac_addrs_dst": 0,
"erspan": 0,
"nsh": 0,
"event": {
  "ipv4": {
    "pkt_too_small": 0,
    "hlen_too_small": 0,
    "iplen_smaller_than_hlen": 0,
    "trunc_pkt": 0,
    "opt_invalid": 0,
    "opt_invalid_len": 0,
    "opt_malformed": 0,
    "opt_pad_required": 0,
    "opt_eol_required": 0,
    "opt_duplicate": 0,
    "opt_unknown": 0,
    "wrong_ip_version": 0,
    "icmpv6": 0,
    "frag_pkt_too_large": 0,
    "frag_overlap": 0,
    "frag_ignored": 0
  },
  "icmpv4": {
    "pkt_too_small": 0,
    "unknown_type": 0,
    "unknown_code": 0,
    "ipv4_trunc_pkt": 0,
    "ipv4_unknown_ver": 0
  },
  "icmpv6": {
    "unknown_type": 0,
    "unknown_code": 0,
    "pkt_too_small": 0,
    "ipv6_unknown_version": 0,
    "ipv6_trunc_pkt": 0,
    "mld_message_with_invalid_hl": 0,
    "unassigned_type": 0,
    "experimentation_type": 0
  },
  "ipv6": {
    "pkt_too_small": 0,
    "trunc_pkt": 0,
    "trunc_exthdr": 0,
    "exthdr_dupl_fh": 0,
    "exthdr_useless_fh": 0,
    "exthdr_dupl_rh": 0,
    "exthdr_dupl_hh": 0,
    "exthdr_dupl_dh": 0,
    "exthdr_dupl_ah": 0,
    "exthdr_dupl_eh": 0,
    "exthdr_invalid_optlen": 0,
    "wrong_ip_version": 0,
    "exthdr_ah_res_not_null": 0,
    "hopopts_unknown_opt": 0,
    "hopopts_only_padding": 0,
    "dstopts_unknown_opt": 0,
    "dstopts_only_padding": 0,
    "rh_type_0": 0,
    "zero_len_padn": 0,
    "fh_non_zero_reserved_field": 0,
    "data_after_none_header": 0,
    "unknown_next_header": 0,
    "icmpv4": 0,
    "frag_pkt_too_large": 0,
    "frag_overlap": 0,
    "frag_invalid_length": 0,
  "frag_ignored": 0,
  "ipv4_in_ipv6_too_small": 0,
  "ipv4_in_ipv6_wrong_version": 0,
  "ipv6_in_ipv6_too_small": 0,
  "ipv6_in_ipv6_wrong_version": 0
},
"tcp": {
  "pkt_too_small": 0,
  "hlen_too_small": 0,
  "invalid_optlen": 0,
  "opt_invalid_len": 0,
  "opt_duplicate": 0
},
"udp": {
  "pkt_too_small": 0,
  "hlen_too_small": 0,
  "hlen_invalid": 0
},
"sll": {
  "pkt_too_small": 0
},
"ethernet": {
  "pkt_too_small": 0
},
"ppp": {
  "pkt_too_small": 0,
  "vju_pkt_too_small": 0,
  "ip4_pkt_too_small": 0,
  "ip6_pkt_too_small": 0,
  "wrong_type": 0,
  "unsup_proto": 0
},
"pppoe": {
  "pkt_too_small": 0,
  "wrong_code": 0,
  "malformed_tags": 0
},
"gre": {
  "pkt_too_small": 0,
  "wrong_version": 0,
  "version0_recur": 0,
  "version0_flags": 0,
  "version0_hdr_too_big": 0,
  "version0_malformed_sre_hdr": 0,
  "version1_chksum": 0,
  "version1_route": 0,
  "version1_ssr": 0,
  "version1_recur": 0,
  "version1_flags": 0,
  "version1_no_key": 0,
  "version1_wrong_protocol": 0,
  "version1_malformed_sre_hdr": 0,
  "version1_hdr_too_big": 0
},
"vlan": {
  "header_too_small": 0,
  "unknown_type": 0,
  "too_many_layers": 0
},
"ieee8021ah": {
  "header_too_small": 0
},
"vntag": {
  "header_too_small": 0,
  "unknown_type": 0
},
"ipraw": {
  "invalid_ip_version": 0
},
"ltnull": {
  "pkt_too_small": 0,
  "unsupported_type": 0
},
"sctp": {
  "pkt_too_small": 0
},
"esp": {
  "pkt_too_small": 0
},
"mpls": {
  "header_too_small": 0,
  "pkt_too_small": 0,
  "bad_label_router_alert": 0,
  "bad_label_implicit_null": 0,
  "bad_label_reserved": 0,
  "unknown_payload_type": 0
},
"vxlan": {
  "unknown_payload_type": 0
},
"geneve": {
  "unknown_payload_type": 0
},
"erspan": {
  "header_too_small": 0,
  "unsupported_version": 0,
        "too_many_vlan_layers": 0
      },
      "dce": {
        "pkt_too_small": 0
      },
      "chdlc": {
        "pkt_too_small": 0
      },
      "nsh": {
        "header_too_small": 0,
        "unsupported_version": 0,
        "bad_header_length": 0,
        "reserved_type": 0,
        "unsupported_type": 0,
        "unknown_payload": 0
      }
    },
    "too_many_layers": 0
  },
  "flow": {
    "memcap": 0,
    "total": 0,
    "active": 0,
    "tcp": 0,
    "udp": 0,
    "icmpv4": 0,
    "icmpv6": 0,
    "tcp_reuse": 0,
    "get_used": 0,
    "get_used_eval": 0,
    "get_used_eval_reject": 0,
    "get_used_eval_busy": 0,
    "get_used_failed": 0,
    "wrk": {
      "spare_sync_avg": 0,
      "spare_sync": 0,
      "spare_sync_incomplete": 0,
      "spare_sync_empty": 0,
      "flows_evicted_needs_work": 0,
      "flows_evicted_pkt_inject": 0,
      "flows_evicted": 0,
      "flows_injected": 0,
      "flows_injected_max": 0
    },
    "end": {
      "state": {
        "new": 0,
        "established": 0,
        "closed": 0,
        "local_bypassed": 0
      },
      "tcp_state": {
        "none": 0,
        "syn_sent": 0,
        "syn_recv": 0,
        "established": 0,
        "fin_wait1": 0,
        "fin_wait2": 0,
        "time_wait": 0,
        "last_ack": 0,
        "close_wait": 0,
        "closing": 0,
        "closed": 0
      },
      "tcp_liberal": 0
    },
    "mgr": {
      "full_hash_pass": 15,
      "rows_per_sec": 6553,
      "rows_maxlen": 0,
      "flows_checked": 0,
      "flows_notimeout": 0,
      "flows_timeout": 0,
      "flows_timeout_inuse": 0,
      "flows_evicted": 0,
      "flows_evicted_needs_work": 0
    },
    "spare": 10000,
    "emerg_mode_entered": 0,
    "emerg_mode_over": 0,
    "recycler": {
      "recycled": 0,
      "queue_avg": 0,
      "queue_max": 0
    },
    "memuse": 7394304
  },
  "tcp": {
    "active_sessions": 0,
    "sessions": 0,
    "ssn_memcap_drop": 0,
    "ssn_from_cache": 0,
    "ssn_from_pool": 0,
    "pseudo": 0,
    "pseudo_failed": 0,
    "invalid_checksum": 0,
      "no_flow": 0,
      "syn": 0,
      "synack": 0,
      "rst": 0,
      "midstream_pickups": 0,
      "pkt_on_wrong_thread": 0,
      "segment_memcap_drop": 0,
      "segment_from_cache": 0,
      "segment_from_pool": 0,
      "stream_depth_reached": 0,
      "reassembly_gap": 0,
      "overlap": 0,
      "overlap_diff_data": 0,
      "insert_data_normal_fail": 0,
      "insert_data_overlap_fail": 0,
      "memuse": 7274496,
      "reassembly_memuse": 1376256
    },
    "defrag": {
      "ipv4": {
        "fragments": 0,
        "reassembled": 0,
        "timeouts": 0
      },
      "ipv6": {
        "fragments": 0,
        "reassembled": 0,
        "timeouts": 0
      },
      "max_frag_hits": 0
    },
    "flow_bypassed": {
      "local_pkts": 0,
      "local_bytes": 0,
      "local_capture_pkts": 0,
      "local_capture_bytes": 0,
      "closed": 0,
      "pkts": 0,
      "bytes": 0
    },
    "detect": {
      "engines": [
        {
          "id": 0,
          "last_reload": "2023-02-01T01:00:45.123456+00:00",
          "rules_loaded": 21944,
          "rules_failed": 2
        }
```

```
  ],
  "alert": 0,
  "alert_queue_overflow": 0,
  "alerts_suppressed": 0
},
"app_layer": {
  "flow": {
    "http": 0,
    "ftp": 0,
    "smtp": 0,
    "tls": 0,
    "ssh": 0,
    "imap": 0,
    "smb": 0,
    "dcerpc_tcp": 0,
    "dns_tcp": 0,
    "nfs_tcp": 0,
    "ntp": 0,
    "ftp-data": 0,
    "tftp": 0,
    "ike": 0,
    "krb5_tcp": 0,
    "quic": 0,
    "dhcp": 0,
    "snmp": 0,
    "sip": 0,
    "rfb": 0,
    "mqtt": 0,
    "telnet": 0,
    "rdp": 0,
    "http2": 0,
    "failed_tcp": 0,
    "dcerpc_udp": 0,
    "dns_udp": 0,
    "nfs_udp": 0,
    "krb5_udp": 0,
    "failed_udp": 0
  },
  "tx": {
    "http": 0,
    "ftp": 0,
    "smtp": 0,
    "tls": 0,
    "ssh": 0,
    "imap": 0,
    "smb": 0,
    "dcerpc_tcp": 0,
    "dns_tcp": 0,
    "nfs_tcp": 0,
    "ntp": 0,
    "ftp-data": 0,
    "tftp": 0,
    "ike": 0,
    "krb5_tcp": 0,
    "quic": 0,
    "dhcp": 0,
    "snmp": 0,
    "sip": 0,
    "rfb": 0,
    "mqtt": 0,
    "telnet": 0,
    "rdp": 0,
    "http2": 0,
    "dcerpc_udp": 0,
    "dns_udp": 0,
    "nfs_udp": 0,
    "krb5_udp": 0
  },
  "error": {
    "http": {
      "gap": 0,
      "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "ftp": {
      "gap": 0,
      "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "smtp": {
      "gap": 0,
      "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "tls": {
      "gap": 0,
      "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "ssh": {
      "gap": 0,
      "alloc": 0,
    "parser": 0,
    "internal": 0
  },
  "imap": {
    "gap": 0,
    "alloc": 0,
    "parser": 0,
    "internal": 0
  },
  "smb": {
    "gap": 0,
    "alloc": 0,
    "parser": 0,
    "internal": 0
  },
  "dcerpc_tcp": {
    "gap": 0,
    "alloc": 0,
    "parser": 0,
    "internal": 0
  },
  "dns_tcp": {
    "gap": 0,
    "alloc": 0,
    "parser": 0,
    "internal": 0
  },
  "nfs_tcp": {
    "gap": 0,
    "alloc": 0,
    "parser": 0,
    "internal": 0
  },
  "ntp": {
    "gap": 0,
    "alloc": 0,
    "parser": 0,
    "internal": 0
  },
  "ftp-data": {
    "gap": 0,
    "alloc": 0,
    "parser": 0,
    "internal": 0
  },
  "tftp": {
    "gap": 0,
    "alloc": 0,
  "parser": 0,
  "internal": 0
},
"ike": {
  "gap": 0,
  "alloc": 0,
  "parser": 0,
  "internal": 0
},
"krb5_tcp": {
  "gap": 0,
  "alloc": 0,
  "parser": 0,
  "internal": 0
},
"quic": {
  "gap": 0,
  "alloc": 0,
  "parser": 0,
  "internal": 0
},
"dhcp": {
  "gap": 0,
  "alloc": 0,
  "parser": 0,
  "internal": 0
},
"snmp": {
  "gap": 0,
  "alloc": 0,
  "parser": 0,
  "internal": 0
},
"sip": {
  "gap": 0,
  "alloc": 0,
  "parser": 0,
  "internal": 0
},
"rfb": {
  "gap": 0,
  "alloc": 0,
  "parser": 0,
  "internal": 0
},
"mqtt": {
  "gap": 0,
  "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "telnet": {
      "gap": 0,
      "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "rdp": {
      "gap": 0,
      "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "http2": {
      "gap": 0,
      "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "failed_tcp": {
      "gap": 0
    },
    "dcerpc_udp": {
      "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "dns_udp": {
      "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "nfs_udp": {
      "alloc": 0,
      "parser": 0,
      "internal": 0
    },
    "krb5_udp": {
      "alloc": 0,
      "parser": 0,
      "internal": 0
    }
  },
  "expectations": 0
},
"http": {
        "memuse": 0,
        "memcap": 0
      },
      "ftp": {
        "memuse": 0,
        "memcap": 0
      },
      "file_store": {
        "open_files": 0
      }
    }
  },
  "ft_stats": {
    "memif_bytes_dropped": 5990,
    "memif_bytes_sent": 21282,
    "memif_packets_dropped": 30,
    "memif_packets_sent": 246,
    "packets_received": 276,
    "bytes_received": 27272
  },
  "interface_stats": {
    "interfaces": [
      {
        "byte_dropped": 0,
        "byte_received": 0,
        "byte_received_perf": 0,
        "ierrors": 0,
        "imissed": 0,
        "ipackets": 0,
        "mbuf_9k_alloc_failed": 0,
        "mbuf_9k_alloc_ok": 0,
        "mbuf_9k_populate_failed": 0,
        "name": "eth0",
        "no_packet": 0,
        "packet_dropped": 0,
        "packet_received": 0,
        "packet_received_perf": 0,
        "rx_loop": 0,
        "rx_nombuf": 0,
        "speed": 0,
        "status": "down",
        "tx_attempt": 0,
        "tx_loop": 0,
        "tx_packet_dropped": 0,
        "type": "ixgbe"
      },
      {
        "byte_dropped": 0,
    "byte_received": 0,
    "byte_received_perf": 0,
    "ierrors": 0,
    "imissed": 0,
    "ipackets": 0,
    "mbuf_9k_alloc_failed": 0,
    "mbuf_9k_alloc_ok": 0,
    "mbuf_9k_populate_failed": 0,
    "name": "eth1",
    "no_packet": 0,
    "packet_dropped": 0,
    "packet_received": 0,
    "packet_received_perf": 0,
    "rx_loop": 0,
    "rx_nombuf": 0,
    "speed": 0,
    "status": "down",
    "tx_attempt": 0,
    "tx_loop": 0,
    "tx_packet_dropped": 0,
    "type": "ixgbe"
  },
  {
    "byte_dropped": 0,
    "byte_received": 27272,
    "byte_received_perf": 0,
    "ierrors": 0,
    "imissed": 0,
    "ipackets": 276,
    "mbuf_9k_alloc_failed": 0,
    "mbuf_9k_alloc_ok": 0,
    "mbuf_9k_populate_failed": 0,
    "name": "eth2",
    "no_packet": 9129055228,
    "packet_dropped": 0,
    "packet_received": 276,
    "packet_received_perf": 0,
    "rx_loop": 9129055499,
    "rx_nombuf": 0,
    "speed": 1000,
    "status": "up",
    "tx_attempt": 0,
    "tx_loop": 0,
    "tx_packet_dropped": 0,
    "type": "dpdk"
  },
  {
    "byte_dropped": 0,
            "byte_received": 0,
            "byte_received_perf": 0,
            "ierrors": 0,
            "imissed": 0,
            "ipackets": 0,
            "mbuf_9k_alloc_failed": 0,
            "mbuf_9k_alloc_ok": 0,
            "mbuf_9k_populate_failed": 0,
            "name": "eth3",
            "no_packet": 0,
            "packet_dropped": 0,
            "packet_received": 0,
            "packet_received_perf": 0,
            "rx_loop": 0,
            "rx_nombuf": 0,
            "speed": 0,
            "status": "down",
            "tx_attempt": 0,
            "tx_loop": 0,
            "tx_packet_dropped": 0,
            "type": "dpdk"
          }
        ]
      },
      "forwarder_stats": {
        "events_processed": 0,
        "events_sent": 0,
        "alerts_dropped_for_rate_limit": 0,
        "alerts_dropped_for_rate_limit_this_hour": 0
      },
      "device_serial": "U12312300000123",
      "timestamp": "2023-02-01T01:23:45.123456+00:00"
    }
  ]
}
```

Vectra Match Status

GET

Used to retrieve Vectra Match status. Returned fields include:

- device_serial: serial of the device for which the information is valid
- is_enabled: boolean of whether Vectra Match is enabled on given device
- process_health: returns “healthy” if process is healthy, else gives description of current process health
- timestamp: timestamp of when the information was gathered
- process_error_detail: returns empty string when healthy, otherwise gives more information about cause of health degredation

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/status`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"

Optional Query Params:
{“device_serial”:”<device_serial>”}
```

Response:
Status Code: 200
```
{
  "status": [
    {
      "device_serial": "U12312300000123",
      "is_enabled": true,
      "process_health": "healthy",
      "timestamp": "2023-02-01T01:23:45.123456+00:00"
    },
    {
      "device_serial": "U12312300000456",
      "is_enabled": true,
      "process_health": "healthy",
      "timestamp": "2023-02-01T01:23:46.234567+00:00"
    },
    {
      "device_serial": "U12312300000789",
      "is_enabled": true,
      "process_health": "healthy",
      "timestamp": "2023-02-01T01:23:47.345678+00:00"
    }
  ]
}
```

Vectra Match Available Devices

GET

Used to retrieve devices on which Vectra Match can be enabled. Returned fields include:

- device_serial: serial of the device
- ip_address: IP address of the device
- is_virtual: boolean, returns True if the device is virtual
- last_seen: ISO-8601 timestamp of the time this device was last seen by the headend
- location: string, user-provided location
- product_name: string, name of the product platform of the device
- version: string, version of Vectra software that the device is currently running
- alias: string, user-provided name for the device
- headend_uri: string, if the device is a sensor, this is the URI of the headend to which it is paired

- mode: string, ‘sensor’ if the device is a sensor, ‘mixed’ if the device is a headend in mixed mode

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/available-devices`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:
Status Code: 200
```
{
  "devices": [
    {
      "device_serial": "U12312300000123",
      "ip_address": "192.168.0.10",
      "is_virtual": false,
      "product_name": "X29v2",
      "version": "7.5.0-1-9036",
      “mode”: “mixed”,
    },
    {
      "alias": "Sensor-Office",
      "device_serial": "U12312300000456",
      "headend_uri": "x29v2-5-10.example.local",
      "ip_address": "192.168.1.10",
      "is_virtual": false,
      "last_seen": "2023-02-01T12:34:56Z",
      "location": "Main Office",
      "product_name": "S101",
      "version": "7.5.0-1-9036",
      “mode”: “sensor”,
    },
    {
      "alias": "Sensor-DC",
      "device_serial": "U12312300000789",
      "headend_uri": "x29v2-5-10.example.local",
      "ip_address": "192.168.2.10",
      "is_virtual": false,
      "last_seen": "2023-02-01T01:23:45Z",
      "location": "New Office",
      "product_name": "S11",
      "version": "7.5.0-1-9036",
      “mode”: “sensor”,
    }
  ]
}
```

Vectra Match Rules

GET

Used to retrieve information about a Vectra Match ruleset, given its UUID

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/rules`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

```
Query Params:
{“uuid”:”<uuid>”}
```

Response:
Status Code: 200

```
{
  "device_serials": [],
  "hashsum": "e5227116937f484e3044ba1f28d6053edb33fc565581d0fda73b587d6a2bfed9",
  "name": "valid_rules.rules",
  "notes": {
    "creator": "vectra",
    "description": "test note"
  },
  "timestamp": "2023-02-01T01:23:45.123456+00:00",
  "uuid": "6aea451c-4156-4ac6-bbeb-c9e0b16b1da5"
}
```

POST

Used to upload a new Vectra Match ruleset. A valid Vectra Match license is required to use this endpoint.

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/rules`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

```
Request cURL Example:

curl --header "Authorization: Token <token>" --insecure -F file=@valid_rules.rules -F
notes="This is a new file" -F rotate=true https://brain.ip/api/v2.5/vectra-match/rules

Request Python Requests Example:

resp = requests.post("https://brain.ip/api/v2.5/vectra-match/rules",
headers={"Authorization":"Token <token>"}, verify=False,
files={"file":open("valid_rules.rules","rb")}, data={"notes":"This is a new file",
“rotate”:”true”})
```

Response:

```
Status code: 200
{
  "device_serials": [],
  "hashsum": "e5227116937f484e3044ba1f28d6053edb33fc565581d0fda73b587d6a2bfed9",
  "name": "valid_rules.rules",
  "notes": {
    "creator": "vectra",
    "description": "test note"
  },
  "timestamp": "2023-02-01T01:23:45.123456+00:00",
  "uuid": "6aea451c-4156-4ac6-bbeb-c9e0b16b1da5"
}
```

DELETE

Used to delete a Vectra Match ruleset

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/rules`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

```
Required Request JSON Body:
{“uuid”:”<uuid>”}
```

Response:

```
Status code: 200
{
  "details": "success"
}
```

Vectra Match Assignment

GET

Used to retrieve information about current Vectra Match ruleset assignments.

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/assignment`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

Response:

Status Code: 200

```
{
  "device_to_rules_map": [
    {
      "device_serial": "U12312300000123",
      "rules": [
        "6aea451c-4156-4ac6-bbeb-c9e0b16b1da5"
      ],
      "timestamp": "2023-02-01T01:23:45.123456+00:00",
      "updated_by": "vadmin"
    },
    {
      "device_serial": "U12312300000456",
      "rules": [
        "6aea451c-4156-4ac6-bbeb-c9e0b16b1da5"
      ],
      "timestamp": "2023-02-01T01:23:45.123456+00:00",
      "updated_by": "vadmin"
    },
    {
      "device_serial": "U12312300000789",
      "rules": [
        "6aea451c-4156-4ac6-bbeb-c9e0b16b1da5"
      ],
      "timestamp": "2023-02-01T01:23:45.123456+00:00",
      "updated_by": "vadmin"
    }
  ],
  "rules_to_devices_map": [
    {
      "device_serials": [
        "U12312300000123",
        "U12312300000456",
        "U12312300000789"
      ],
      "hashsum": "e5227116937f484e3044ba1f28d6053edb33fc565581d0fda73b587d6a2bfed9",
      "name": "valid_rules.rules",
      "notes": {
        "creator": "vectra",
        "description": "test note"
      },
      "timestamp": "2023-02-01T01:23:45.123456+00:00",
      "uuid": "6aea451c-4156-4ac6-bbeb-c9e0b16b1da5"
    }
  ]
}
```

POST

Used to assign a Vectra Match ruleset to one or more devices. A valid Vectra Match license is required to use this endpoint.

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/assignment`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

```
Required Request JSON Body:
{“uuid”:”<uuid>”, “device_serials”:[“<serial1>”, “<serial2>”,…]}
```

Response:
```
Status code: 200
{
  "details": "success"
}
```

DELETE

Used to remove a rules file from a given device

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/assignment`

Headers:
```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"
```

```
Required Request JSON Body:
{“uuid”:”<uuid>”, “device_serial”:”<device_serial>”}
```

Response:
```
Status code: 200
{
  "details": "success"
}
```

Vectra Match Alert Stats

GET

Used to retrieve Vectra Match alert stats. Returned fields include:

- top_alert_counts: statistics directly pulled from the Vectra Match (suricata) engine about the top 10 alerts that have been processed since the last rotation of the suricata’s eve.json event log
- eve_log_rotated_time: ISO-8601 timestamp of the last time the eve.json log was rotated, which helps to indicate the time window the alert counts were collected over (from this timestamp until the current time)
- device_serial: serial of the device for which the stats are given

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/alert-stats`

Headers:

```
“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"

Optional Query Params:
{“device_serial”:”<device_serial>”}
```

Response:
Status Code: 200

```
{
    "alert_stats": [
        {
            "top_alert_counts": [
                {"count": 1032, "signature_id": 2100498, "signature": "GPL
ATTACK_RESPONSE id check returned root"},
                {
                    "count": 101,
                    "signature_id": 2006380,
                    "signature": "ET POLICY Outgoing Basic Auth Base64 HTTP Password
detected unencrypted",
                },
                {"count": 78, "signature_id": 2028807, "signature": "ET JA3 Hash -
[Abuse.ch] Possible Tofsee"},
                {"count": 60, "signature_id": 2034636, "signature": "ET INFO Python
SimpleHTTP ServerBanner"},
                {
                    "count": 34,
                    "signature_id": 2001580,
                    "signature": "ET SCAN Behavioral Unusual Port 137 traffic
Potential Scan or Infection",
                },
                {"count": 27, "signature_id": 2028763, "signature": "ET JA3 Hash -
[Abuse.ch] Possible Adwind"},
                {"count": 12, "signature_id": 2017398, "signature": "ET POLICY IP
Check Domain (icanhazip. com in HTTP Host)"},
                {"count": 10, "signature_id": 2027390, "signature": "ET USER_AGENTS
Microsoft Device Metadata Retrieval Client User-Agent"},
                {"count": 8, "signature_id": 2035466, "signature": "ET INFO Observed
Discord Domain in DNS Lookup (discordapp .com)"},
                {"count": 7, "signature_id": 2016922, "signature": "ET MALWARE
Backdoor family PCRat/Gh0st CnC traffic"},
            ],
            "device_serial": "A21010000000002",
            "eve_log_rotated_time": "2023-03-23T18:00:00",
        }
    ]
}
```

Vectra Match Download Vectra Ruleset

GET

Used to download Vectra curated ruleset.

Note: When using cURL the ‘-o’ option will need to be used to redirect the output to the location you wish to save the file.

URL: `https://<vectra_management_ip>/api/v2.5/vectra-match/download-vectra-ruleset`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

```
Request cURL Example:

curl -o curated.rules --header "Authorization: Token <token>" --insecure
https://brain.ip/api/v2.5/vectra-match/download-vectra-ruleset
```

Response:
Status Code: 200
`Streams contents of curated.rules file`

Health Events

GET

URL: `https://<vectra_management_ip>/api/v2.5/events/health`

Headers:
`“Authorization”: “Token <api-key>” OR "Authorization": "Bearer <access_token>"`

Response:
```
{
 "events": [
  {
   "event_object": {
    "type": "device",
    "name": "saas123124"
   },
   "health_check_name": "cpu_check",
   "status": {
    "code": 0,
    "label": "OK"
   },
   "output": "{\"cpu_usage\": 45.2}",
   "event_timestamp": "2024-01-15T10:30:00Z",
   "last_state_change_at": "2024-01-15T09:30:00Z",
   "last_state": "OK",
      "last_ok_at": "2024-01-15T10:00:00Z"
    },
    {
      "event_object": {
        "type": "brain",
        "name": "test_brain_1"
      },
      "health_check_name": "memory_check",
      "status": {
        "code": 1,
        "label": "WARNING"
      },
      "output": "{\"memory_usage\": 85.7}",
      "event_timestamp": "2024-01-15T10:25:00Z",
      "last_state_change_at": "2024-01-15T08:30:00Z",
      "last_state": "WARNING",
      "last_ok_at": "2024-01-15T07:00:00Z"
    }
  ],
  "next_checkpoint": 1234,
  "remaining_count": 1
}
```

Appendix B

Detections

Predefined categories and vnames

CATEGORY	VNAME
Command and Control	
	External Remote Access
	Hidden DNS Tunnel
	Hidden HTTP Tunnel
	Hidden HTTPS Tunnel
	Malware Update
	Peer to Peer
	Pulling Instructions
	Stealth HTTP Post
	Suspect Domain Activity
	Suspicious HTTP
	TOR Activity
	Threat Intelligence Match
	Suspicious Relay
	Multi-home Fronted Tunnel
Botnet	
	Abnormal Ad Activity
	Abnormal Web Activity
	Cryptocurrency Mining
	Brute-Force
	Outbound DoS
	Outbound Port Sweep
	Outbound Spam
Reconnaissance	
	File Share Enumeration
	Internal Darknet Scan
	Kerberos Account Scan
	Port Scan
	Port Sweep
	SMB Account Scan

CATEGORY	VNAME
	RDP Recon
	RPC Recon
	Suspicious LDAP Query
Lateral Movement	
	Automated Replication
	Brute-Force
	Kerberos Brute-Force
	Kerberos Server Access
	Privilege Anomaly: Unusual Account on Host
	Privilege Anomaly: Unusual Host
	Privilege Anomaly: Unusual Service
	Privilege Anomaly: Unusual Service from Host
	Privilege Anomaly: Unusual Trio
	Ransomware File Activity
	Shell Knocker Client
	Shell Knocker Server
	SMB Brute-Force
	SQL Injection Activity
	Suspicious Admin
	Suspicious Kerberos Account
	Suspicious Kerberos Client
	Suspicious Remote Desktop
	Suspicious Remote Execution
	Threat Intelligence Match
Exfiltration	
	Data Smuggler
	Hidden DNS Tunnel
	Hidden HTTP tunnel
	Hidden HTTPS tunnel
	Smash and Grab
	Staged Transfer - Hop 1
	Threat Intelligence Match
Info	
	Custom

Settings

Predefined group values.

GROUP POSSIBLE KEY VALUES	DESCRIPTION OF THE VALUES
vectra	Lists the hostname assigned to the Vectra brain
dns	Lists the DNS servers configured for host resolution. Up to 3 servers can be configured.
ntp	Lists the NTP server used for time synchronization. Up to 5 servers can be configured.
syslog	Lists the syslog servers and their configuration used for log forwarding. Up to 6 servers can be configured.
platform	Lists the OS version installed.
digest	Lists the email addresses to forward reports to. Up to 3 email addresses can be added to receive the digest emails.
alert	Lists the email addresses to forward detection alerts to. Up to 3 email addresses can be added to receive the alert emails.
smtp	Lists the SMTP server information used to relay alert and digest emails from.
datasharing	Lists the current setting to share data to the Vectra cloud. The key value is Boolean, either True or False.
update	Lists the timestamp when the last OS update was applied to the system.
feature	Lists the setting for the community feature to be enabled or disabled. The key value is Boolean, either On or Off.