EclecticIQ Endpoint Security Platform (ESP)

REST API Documentation

Table of Contents

Use Cases:

- Endpoint Node Information and Management
- Tagging and Logical Grouping of Endpoints
- Scheduled Queries, Distributed (Ad-Hoc) Queries
- Rules and Alerts
- Active Response

1. Overview

The EclecticIQ Endpoint Security Platform (ESP) is a combination of endpoint agents, an endpoint fleet manager.

EclecticIQ ESP REST API allows developers to use a programming language of their choice to integrate with the headless EclecticIQ ESP server. The REST APIs provide the means to configure and query the data from the fleet manager. All payloads are exchanged over REST and use the JSON schema.

2. REST Based API

--> Makes use of standard HTTP verbs like GET, POST, DELETE.

--> Uses standard HTTP error responses to describe errors

--> Authentication provided using API keys in the HTTP Authorization header

--> Requests and responses are in JSON format.

3. Versioning

The EclecticIQ ESP API is a versioned API. We reserve the right to add new parameters, properties, or resources to the API without advance notice. These updates are considered non-breaking, and the compatibility rules below should be followed to ensure your application does not break.

Breaking changes such as removing or renaming an attribute will be released as a new version of the API. EclecticIQ will provide a migration path for new versions of APIs and will communicate timelines for end-of-life when deprecating APIs. Do not consume any API unless it is formally documented. All undocumented endpoints should be considered private, subject to change without notice, and not covered by any agreements.

The API version is currently v1. All API requests must use the https scheme.

4. BASE_URL

API calls are made to a URL to identify the location from which the data is accessed. You must replace the placeholders <server IP> with actual details for your EclecticIQ ESP server. The BASE_URL follows this template: *https://<server_ip>/esp-ui/services/api/v1*

5. Authentication

The EclecticIQ ESP API requires all requests to present a valid API key (x-access-token: API Key) specified in the HTTP Authorization header for every HTTP request. While logging in (*https://<BASE_URL>/login*) the x-access-token will be provided from the server, which needs to be used for further API calls. If the API key is missing or invalid, a 401 unauthorized response code is returned.

The API key (x-access-token) has the privileges associated with an administrator account. The API key cannot be used to authenticate once the user logs out from the platform or the default expiry time (7 days is set for now) is reached. If you believe your API key is compromised/expired, you can generate a new one. This ensures that the older API key can no longer be used to authenticate to the server.

x-access-token:

The EclecticIQ ESP server provides an auth token called x-access-token, which is encoded JWT and is used as a unique key for all API calls further. x-access-token will be provided at the URL *https://<BASE_URL>/login*.

6. Transport Security

HTTP over TLS v1.2 is enforced for all API calls. Any non-secure calls will be rejected by the server.

7. Client Request Context

EclecticIQ ESP will derive client request context directly from the HTTP request headers and client TCP socket. Request context is used to evaluate policies and provide client information for troubleshooting and auditing purposes.

User Agent: EclecticIQ ESP supports the standard User-Agent HTTP header to identify the client application. Always send a User-Agent string to uniquely identify your client application and version such as SOC Application/1.1.

IP Address: The IP address of your application will be automatically used as the client IP address for your request.

8. Errors

All requests on success will return a 200 status if there is content to return or a 204 status if there is no content to return. HTTP response codes are used to indicate API errors.

Code	Description
400	Malformed or bad JSON Request
401	API access without authentication or invalid API key
404	Resource not found.
410	Resource gone.
422	Request can be parsed. But, has invalid content.
429	Too many requests. Server has encountered rate limits.
200	Success
201	Created. Returns after a successful POST when a resource is created.
500	Internal server error
503	Service currently unavailable

9. Request Debugging

The Auth Token(x-access-token) will always be present in every API response and can be used for debugging. The following header is set in each response.

x-access-token - The unique identifier for the API request.

HTTP / 1.1 200 OK

{

x-access-token:

”eyJhbGciOiJIUzUxMiIsImlhdCI6MTU2NzY3MjcyMCwiZXhwIjoxNTY3NjczMzIwfQ.eyJpZCI6MX0.7jklhAly5ZO6xr1t0Y2ahkZvEEMnrescGK9nszqF-hMAProwbjOHaiRO3tBS5I2gdmVSqKqBHynvmor7TA”

}

10. Terminology

Fleet	Set of endpoints running the EclecticIQ ESP agent and managed by the EclecticIQ ESP server
Node	A specific endpoint that is actively monitored
Config	EclecticIQ ESP OS Query based agent derives its behaviour from its configuration. The config is a JSON describing the assorted options used to instrument the agent behaviour as well as the queries scheduled on the agent. Config is applied at a node level. Refer the product guide for supported configurations.
Options	Options (or flags) are the set of parameters the agent uses to affect its behaviour. A list of all the flags supported can be found at *https://osquery.readthedocs.io/en/stable/installation/cli-flags/* Options can be retrieved as part of config.
Tag	A mechanism to logically group/associate elements such as nodes, packs etc.
Scheduled Query	Queries that run on a specified scheduled on an endpoint
Query Pack	Grouping of scheduled queries
Ad Hoc Query	A live, on-demand query that is targeted at an endpoint or a set of endpoints. Also referred to as a distributed query.
Alerts	Rules can be applied to results of scheduled queries. When events match with a rule, the EclecticIQ ESP server can generate an alert with the event information for proactive analysis by the SOC analyst.
Active Response	Actions that be taken on affected endpoint(s) as part of Incident Response activity.

REST API Section:

Headers required:

For POST Method (except /login):

{

“Content-Type”: ”application/json”,

“x-access-token”: ”<received from /login Api>”

}

For GET Method:

{

“x-access-token”:” <received from /login Api>”

}

For /login API:

{

“Content-Type”:”application/json”

}

For all (GET/POST) APIs with File Uploads:

{

“Content-Type”: ”multipart/form-data”,

“x-access-token”: ”<received from /login api>”

}

Get a file from downloads path.

Returns a response of a file object from downloads path for a specific path given.

URL: *https://<SERVER-IP>/downloads/<path: filename>*

Or https://<SERVER-IP>:9000/downloads/<path: filename>

Request type: GET

https://<SERVER-IP>/downloads/certificate.crt -- to download the certificate.

https://<SERVER-IP>/downloads/windows/plgx_cpt.exe -- to download the windows Client Provisioning Tool

https://<SERVER-IP>/downloads/linux/plgx_cpt -- to download the Linux Client Provisioning Tool

https://<SERVER-IP>/downloads/plgx_cpt.sh -- to download the mac installer.

BLUEPRINT: General APIs

Blueprint-path: /

User's login

Returns an auth token for the user to authenticate with.

URL: *https://<BASE_URL>/login*

Request type: POST

Example payload format:

```
{
    "username": "admin",
    "password": "admin"
}
```

Required payload arguments: username and password.

Response: Returns a JSON array of JWT auth token (x-access-token).

Example response format:

```
{
    "x-access-token":
    "eyJhbGciOiJIUzUxMiIsImlhdCI6MTU2NzY3MjcyMCwiZXhwIjoxNTY3NjczMzIwfQ.eyJpZCI6MX
    0.7jklhAly5ZO6xr1t0Y2ahkZvEEMnrescGK9nszqF-
    hMAProwbjOHaiRO3tBS5I2gdmVSqKqBHynveAFbmor7TA "
}
```

User's password change

Changes user's password.

URL: *https://<BASE_URL>/management/changepw*

Request type: POST

Example payload format:

```
{
    "old_password": "admin",
    "new_password": " admin123",
    "confirm_new_password": "admin123"
}
```

Required payload arguments: old_password, new_password and confirm_new_password

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": “password is updated successfully “
}
```

User’s password verify.

Verifies user’s password.

URL: *https://<BASE_URL>/management/verifypw*

Request type: POST

Example payload format:

```
{
        "password": "admin"
}
```

Required payload arguments: password.

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": "Password for the current user is verified successfully".
}
```

User’s logout

Makes access token invalid.

URL: *https://<BASE_URL>/logout*

Request type: POST

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
```

```
    "message": “user logged out successfully “
}
```

Update Threat Intel keys

Updates the Threat Intel keys used by EclecticIQ ESP.

URL: *https://<BASE_URL>/management/apikeys*

Request type: POST

Example payload format:

```
{
    "IBMxForceKey": "304020f8-99fd-4a17-9e72-80033278810a",
    "IBMxForcePass": "6710f119-9966-4d94-a7ad-9f98e62373c8",
    "vt_key": "69f922502ee0ea958fa0ead2979257bd084fa012c283ef9540176ce857ac6f2c",
    “otx_key”: ” 69f922502ee0ea958fa0ead2979257bd084fa012c”
}
```

Response: Returns a JSON array of a status, data, and message.

Example response format:

```
{
    "status": "success",
    "message": "Threat Intel keys are updated successfully",
    "data": {
        "ibmxforce": {
            "key": "304020f8-99fd-4a17-9e72-80033278810a",
            "pass": "6710f119-9966-4d94-a7ad-9f98e62373c8"
        },
        "virustotal": {
            "key":
"69f922502ee0ea958fa0ead2979257bd084fa012c283ef9540176ce857ac6f2c"
        },
    }
}
```

View Threat Intel keys

Returns the Threat Intel keys used by EclecticIQ ESP.

URL: *https://<BASE_URL>/management/apikeys*

Request type: GET

Response: Returns a JSON array of a status, data, and message.

Example response format:


```
{
        "status": "success",
        "message": "Threat Intel Keys are fetched successfully",
        "data": {
                "ibmxforce": {
                        "key": "304020f8-99fd-4a17-9e72-80033278810a",
                        "pass": "6710f119-9966-4d94-a7ad-9f98e62373c8"
                },
                "virustotal": {
                        "key":
                "69f922502ee0ea958fa0ead2979257bd084fa012c283ef9540176ce857ac6f2c"
                }
        }
}
```


View Virus Total AV engines configuration

Returns the Virus total anti-virus engines configuration for EclecticIQ ESP use.

URL: *https://<BASE_URL>/management/virustotal/av_engine*

Request type: GET

Response: Returns a JSON array of a status, data, and message.

Example response format:


```
{
        "message": "virus total av engines are fetched successfully",
        "status": "success",
        "data": {
                "min_match_count": 3,
                "av_engines": {
                        "Bkav": {
                                "status": false
```

```
                        },
                        "Sophos ML": {
                                "status": false
                        }
                }
        }
}
```


Update Virus Total AV engines configuration

Updates the Virus total anti-virus engines configuration for EclecticIQ ESP use.

URL: *https://<BASE_URL>/management/virustotal/av_engine*

Request type: POST

Example payload format:


```
{
        "min_match_count": 3,
        "av_engines": {
                "Bkav": {
                        "status": false
                },
                "Sophos ML": {
                        "status": false
                }
        }
}
```


Response: Returns a JSON array of a status, data, and message.

Example response format:


```
{
        "message": "successfully updated av engines",
        "status": "success"
}
```


Hunt through file upload

Hunt on Result Log through the file of indicators uploaded.

URL: *https://<BASE_URL>/hunt-upload*

Request type: POST

Example payload format 1:

```
{
        “file”: <file of indicators>,
        “type”:”md5”
}
```

Required payload arguments: file and type.

Example response format 1:

```
{
         "status": "success",
         "message": "Successfully fetched the results through the hunt",
         "data": [
                  {
                           "hostname": "EC2AMAZ-2RJ1BIF",
                           "host_identifier": "EC2CE2E2-3D74-1248-2FA9-23F2E960ED42",
                           "queries": [
                                    {
                                      "query_name": "osquery_info",
                                      "count": 1
                                    }
                                ]
                  }
         ]
}
```

Example payload format 2:

```
{
        “file”: <file of indicators>,
        “type”:”md5”,
        “host_identifier”:”EC2300D6-B0D5-F9A6-1237-6553106EC525”,
        “query_name”:”win_file_events”,
```

```
        “start”:2,
        “limit”:10
}
```


Required payload arguments: file, type, host_identifier, query_name, start and limit

Example response format 2:


```
{
        "status": "success",
        "message": "Successfully fetched the results through the hunt",
        "data": {
                "count": 1,
                "results": [
                        {
                          "pid": "4752",
                          "uuid": "EC2CE2E2-3D74-1248-2FA9-23F2E960ED42",
                          "version": "4.0.2",
                          "watcher": "-1",
                          "extensions": "active",
                          "start_time": "1592672947",
                          "config_hash": "71f4969da7d79f6b2cbeb64d02e04b17bd8815e7",
                          "instance_id": "78a850bf-844e-426a-8cc6-a66d3975a2ba",
                          "build_distro": "10",
                          "config_valid": "1",
                          "build_platform": "windows"
                        }
                ]
        }
}
```


Hunt through list of indicators

Hunt on Result Log through the list of indicators provided.

URL: *https://<BASE_URL>/indicators/hunt*

Request type: POST

Example payload format 1:

```
{
        “indicators”: “275a71899f7db9d1663fc695ec2fe2a2c4538,
        275a71899fdjsaddb9d1663fc695ec2fe2a2c453fsgs”,
        “type”:”md5”
}
```

Required payload arguments: type and indicators

Example response format 1:

```
{
        "status": "success",
        "message": "Successfully fetched the results through the hunt",
        "data": [
                {
                        "hostname": "EC2AMAZ-2RJ1BIF",
                        "host_identifier": "EC2CE2E2-3D74-1248-2FA9-23F2E960ED42",
                        "queries": [
                                {
                                  "query_name": "osquery_info",
                                  "count": 1
                                }
                            ]
                }
        ]
}
```

Example payload format 2:

```
{
        “indicators”: <file of indicators>,
        “type”:”md5”,
        “host_identifier”:”EC2300D6-B0D5-F9A6-1237-6553106EC525”,
        “query_name”:”win_file_events”,
        “start”:2,
        “limit”:10
```

```
}
```


Required payload arguments: indicators, type, host_identifier, query_name, start and limit

Example response format 2:


```
{
        "status": "success",
        "message": "Successfully fetched the results through the hunt",
        "data": {
                "count": 1,
                "results": [
                        {
                          "pid": "4752",
                          "uuid": "EC2CE2E2-3D74-1248-2FA9-23F2E960ED42",
                          "version": "4.0.2",
                          "watcher": "-1",
                          "extensions": "active",
                          "start_time": "1592672947",
                          "config_hash": "71f4969da7d79f6b2cbeb64d02e04b17bd8815e7",
                          "instance_id": "78a850bf-844e-426a-8cc6-a66d3975a2ba",
                          "build_distro": "10",
                          "config_valid": "1",
                          "build_platform": "windows"
                        }
                ]
        }
}
```


Export Hunt results

Export the hunt results to a csv file.

URL: *https://<BASE_URL>/hunt-upload/export*

Request type: POST

Example payload format:


```
{
```

```
    “file”: <file of indicators>,
    “type”:”md5”,
    “host_identifier”:”EC2300D6-B0D5-F9A6-1237-6553106EC525”,
    “query_name”:”win_file_events”
}
```

Required payload arguments: file, type, host_identifier, query_name

Example response format:

A CSV file object with hunt results.

Search in result log:

Searches for results in Result Log for the conditions given.

URL: *https://<BASE_URL>/search*

Request type: POST

Example payload format 1:

```
{
    "conditions": {
        "condition": "OR",
        "rules": [
            {
                "id": "name",
                "field": "name",
                "type": "string",
                "input": "text",
                "operator": "contains",
                "value": "EC2"
            },
            {
                "id": "name",
                "field": "name",
                "type": "string",
                "input": "text",
                "operator": "equal",
```

```
                    "value": "pc"
                }
            ],
            "valid": true
        }
}
```


Required payload arguments: conditions

Example response format 1:


```
{
    "status": "success",
    "message": "Successfully fetched the results through the payload given",
    "data": [
        {
            "hostname": "EC2AMAZ-2RJ1BIF",
            "host_identifier": "EC2CE2E2-3D74-1248-2FA9-23F2E960ED42",
            "queries": [
                {
                  "query_name": "osquery_info",
                  "count": 1
                }
            ]
        }
    ]
}
```


Example payload format 2:


```
{
    "conditions": {
        "condition": "OR",
        "rules": [
            {
                "id": "name",
                "field": "name",
```

```
                        "type": "string",
                        "input": "text",
                        "operator": "contains",
                        "value": "EC2"
                    },
                    {
                        "id": "name",
                        "field": "name",
                        "type": "string",
                        "input": "text",
                        "operator": "equal",
                        "value": "pc"
                    }
                ],
                "valid": true
            },
            “host_identifier”:”EC241E83-BDC2-CAFC-BF9F-28C22B37A7F0”,
            "query_name":"per_query_perf",
            "start":2,
            "limit":2
}
```


Required payload arguments: conditions, host_identifier, query_name, start and limit

Example response format 2:


```
{
    "status": "success",
    "message": "Successfully fetched the results through the payload given",
    "data": {
        "count": 1,
        "results": [
            {
                "pid": "4752",
                "uuid": "EC2CE2E2-3D74-1248-2FA9-23F2E960ED42",
```

```
                "version": "4.0.2",
                "watcher": "-1",
                "extensions": "active",
                "start_time": "1592672947",
                "config_hash": "71f4969da7d79f6b2cbeb64d02e04b17bd8815e7",
                "instance_id": "78a850bf-844e-426a-8cc6-a66d3975a2ba",
                "build_distro": "10",
                "config_valid": "1",
                "build_platform": "windows"
            }
        ]
    }
}
```


Filter results for indicators uploaded

Filtering Result Log through the file of indicators uploaded for the datetime filters given.

URL: *https://<BASE_URL>/indicators/upload*

Request type: POST

Example payload format:


```
{
    “file”: <file of indicators>,
    “indicator_type”:”md5”,
    “host_identifier”:”EC2300D6-B0D5-F9A6-1237-6553106EC525”,
    “query_name”:”win_file_events”,
    “start”:2,
    “limit”:10,
    “duration”:”3”,
    “date”: "2020-8-5",
    “type”:”2”
}
```


Filters description:

duration – to get recent alerts by(month(4)/week(3)/day(2)/hr(1))

date – end date for the duration to be calculated by(format : 2020-10-14)

type – start date(1)/end date(2)

Required payload arguments: file, type, start and limit

Example response format:


```
{
    "status": "success",
    "message": "Successfully fetched the results through the hunt",
    "data": {
        "count": 28,
        "results": [
            {
                "id": 172270780,
                "name": "process_events",
                "timestamp": "19-10-2020 10:10:47.000000",
                "action": "added",
                "columns": {
                    "cwd": "\"/var/backups\"",
                    "eid": "0000023296",
                    "gid": "0",
                    "pid": "2625",
                    "uid": "0",
                    "auid": "4294967295",
                    "egid": "0",
                    "euid": "0",
                    "path": "/usr/bin/python3.6",
                    "time": "1603102243",
                    "ctime": "1602831478",
                    "parent": "2619",
                    "cmdline": "/usr/bin/python3 -Es /usr/bin/lsb_release -
                i -s"
                },
                "node_id": 60,
                "uuid": "41184ad2-f651-4b9d-baff-f201fc38ce76",
```

```
                                    "status": 0,
                                    "task_id": null,
                                    "hostname": "ip-172-31-29-39",
                                    "host_identifier": "ec21114b-ab50-90fb-02e6-ae03087a3312"
                                }
                            ]
                        }
}
```


Export indicators filtered results

Export the indicators filtered results to a csv file for the datetime filters given.

URL: *https://<BASE_URL>/indicators/upload/export*

Request type: POST

Example payload format:


```
{
        “file”: <file of indicators>,
        “indicator_type”:”md5”,
        “host_identifier”:”EC2300D6-B0D5-F9A6-1237-6553106EC525”,
        “query_name”:”win_file_events”,
        “duration”:”3”,
        “date”: "2020-8-5",
        “type”:”2”
}
```


Filters description:

duration – to get recent alerts by(month(4)/week(3)/day(2)/hr(1))

date – end date for the duration to be calculated by(format : 2020-10-14)

type – start date(1)/end date(2)

Required payload arguments: file, type

Example response format:

A CSV file object with hunt results.

Search in result log for only latest records:

Searches for results in Result Log for the conditions given for latest records only.

URL: *https://<BASE_URL>/activity/search*

Request type: POST

Example payload format:

```
{
        "conditions": {
                "condition": "OR",
                "rules": [
                        {
                                "id": "name",
                                "field": "name",
                                "type": "string",
                                "input": "text",
                                "operator": "contains",
                                "value": "EC2"
                        },
                        {
                                "id": "name",
                                "field": "name",
                                "type": "string",
                                "input": "text",
                                "operator": "equal",
                                "value": "pc"
                        }
                ],
                "valid": true
        },
        “host_identifier”:”EC241E83-BDC2-CAFC-BF9F-28C22B37A7F0”,
        "query_name":"per_query_perf",
        "start":2,
        "limit":2,
        “duration”:”3”,
        “date”: "2020-8-5",
```

```
        “type”:”2”
}
```

Filters description:

duration – to get recent alerts by(month(4)/week(3)/day(2)/hr(1))

date – end date for the duration to be calculated by(format : 2020-10-14)

type – start date(1)/end date(2)

Required payload arguments: conditions, start and limit

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the results through the search",
        "data": {
                "count": 28,
                "results": [
                        {
                                "id": 172270780,
                                "name": "process_events",
                                "timestamp": "19-10-2020 10:10:47.000000",
                                "action": "added",
                                "columns": {
                                        "cwd": "\"/var/backups\"",
                                        "eid": "0000023296",
                                        "gid": "0",
                                        "pid": "2625",
                                        "uid": "0",
                                        "auid": "4294967295",
                                        "egid": "0",
                                        "euid": "0",
                                        "path": "/usr/bin/python3.6",
                                        "time": "1603102243",
                                        "ctime": "1602831478",
                                        "parent": "2619",
```

```
                    "cmdline": "/usr/bin/python3 -Es /usr/bin/lsb_release -i -s"
                },
                "node_id": 60,
                "uuid": "41184ad2-f651-4b9d-baff-f201fc38ce76",
                "status": 0,
                "task_id": null,
                "hostname": "ip-172-31-29-39",
                "host_identifier": "ec21114b-ab50-90fb-02e6-ae03087a3312"
            }
        ]
    }
}
```


Delete recent query result

Deletes the query result for some recent days for the number given.

URL: *https://<BASE_URL>/queryresult/delete*

Request type: POST

Example payload format:


```
{
    “days_of_data”: 2
}
```


Required payload arguments: days_of_data

Response: Returns JSON array of data, status and message

Example response format:


```
{
    "status": "success",
    "message": “Query result data is deleted successfully “,
    “data”:7
}
```


View platform settings

Returns some settings that EclecticIQ ESP uses.

URL: *https://<BASE_URL>/management/settings*

Request type: GET

Response: Returns JSON array of data, status and message

Example response format:

```
{
        "status": "success",
        "message": "Platform settings are fetched successfully",
        "data": {
                "purge_data_duration": 60,
                "alert_aggregation_duration": 60
        }
}
```

Update platform settings

Updates the settings used by EclecticIQ ESP.

URL: *https://<BASE_URL>/management/settings*

Request type: PUT

Example payload format:

```
{
        "purge_data_duration":60,
        "alert_aggregation_duration":60
}
```

Filters description:

purge_data_duration – interval to schedule the data (Alerts, Recent Activity) purge duration by

alert_aggregation_duration – duration by which alerts should be aggregated

Response: Returns JSON array of status and message

Example response format:

```
{
        "status": "success",
        "message": “Platform settings are updated successfully “,
}
```

Export schedule query results

Returns a response of a csv file object with schedule query results.

URL: *https://<BASE_URL>/schedule_query/export*

Request type: POST

Example payload format:

```
{
        “query_name”: “win_registry_events”,
        “host_identifier”:”EC259C26-B72F-553F-A2B3-FD9517DAE7D2”
}
```

Required payload arguments: query_name and host_identifier

Response: Returns a csv file

BLUEPRINT: dashboard

Blueprint-path: /dashboard

Get dashboard data

Get data required for EclecticIQ ESP for dashboard.

URL: *https://<BASE_URL>/dashboard*

Request type: GET

Example response format:

```
{
        "status": "success",
        "message": "Data is fetched successfully",
        "data": {
                "alert_data": {
                        "top_five": {
                                "rule": [
                                        {
                                          "rule_name": "Executable used by PlugX in
                                Uncommon Location",
                                          "count": 4609
```

```
        },
        {
          "rule_name": "test_rule",
          "count": 13
        },
        {
          "rule_name": "Service Stop",
          "count": 2
        }
    ],
    "hosts": [
        {
          "host_identifier": "EC2CD1A0-140B-9331-7A60-
    CFFCE29D2E71",
          "count": 4629
        }
    ],
    "query": [
        {
          "query_name": "Test_query",
          "count": 4609
        },
        {
          "query_name": "win_file_events",
          "count": 14
        },
        {
          "query_name": "win_image_load_events",
          "count": 4
        },
        {
          "query_name": "win_process_events",
          "count": 2
```

```
                }
            ]
        },
        "source": {
            "ioc": {
                "INFO": 0,
                "LOW": 0,
                "WARNING": 0,
                "CRITICAL": 0,
                "TOTAL": 0
            },
            "rule": {
                "INFO": 13,
                "LOW": 0,
                "WARNING": 0,
                "CRITICAL": 4611,
                "TOTAL": 4624
            },
            "virustotal": {
                "INFO": 0,
                "LOW": 5,
                "WARNING": 0,
                "CRITICAL": 0,
                "TOTAL": 5
            },
            "ibmxforce": {
                "INFO": 0,
                "LOW": 0,
                "WARNING": 0,
                "CRITICAL": 0,
                "TOTAL": 0
            },
```

```
                    "alienvault": {
                        "INFO": 0,
                        "LOW": 0,
                        "WARNING": 0,
                        "CRITICAL": 0,
                        "TOTAL": 0
                    }
                }
            },
            "distribution_and_status": {
                "hosts_platform_count": [
                    {
                        "os_name": "ubuntu",
                        "count": 1
                    },
                    {
                        "os_name": "windows",
                        "count": 1
                    }
                ],
                "hosts_status_count": {
                    "online": 2,
                    "offline": 0
                }
            }
        }
}
```


BLUEPRINT: hosts

Blueprint-path: /hosts

Export hosts information

Returns a response of a csv file with all hosts information.

URL: *https://<BASE_URL>/hosts/export*

Request type: GET

Response: Returns a csv file.

View all hosts

Lists all hosts managed by EclecticIQ ESP for the filters applied.

URL: *https://<BASE_URL>/hosts*

Request type: POST

Example payload format:

```
{
        "status":false,
        "platform":"windows",
        “searchterm”:”EC2”,
        "start":0,
        "limit":10,
        “enabled”:true,
        “alerts_count”:true
}
```

Filters description:

status – true – to get all active hosts

status – false – to get all inactive hosts

platform – to filter the results with platform

enabled – true – to get all non-removed hosts

enabled – false – to get all removed hosts

alerts_count– true/false – true to get non resolved alerts count of the host

Response: Returns JSON array of hosts and their properties.

Example response format:

```
{
        "status": "success",
        "message": "Successfully fetched the nodes details",
```

```
"data": {
        "results": [
         {
                "id": 2,
                "display_name": "EC2AMAZ-2RJ1BIF",
                "host_identifier": "EC2CE2E2-3D74-1248-2FA9-23F2E960ED42",
                "os_info": {
                        "name": "Microsoft Windows Server 2019 Datacenter",
                        "build": "17763",
                        "major": "10",
                        "minor": "0",
                        "patch": "",
                        "version": "10.0.17763",
                        "codename": "Server Datacenter (full installation)",
                        "platform": "windows",
                        "install_date": "20190613115936.000000+000",
                        "platform_like": "windows"
                },
                "tags": [
                    "zdsd"
                ],
                "last_ip": "15.206.168.222",
                "alerts_count": 20,
                "is_active": false
         }
        ],
        "count": 3,
        "total_count": 3
    }
}
```

View a host

Lists a node info managed by the EclecticIQ ESP and its properties.

URL: *https://<BASE_URL>/hosts/<string:host_identifier>*

https://<BASE_URL>/hosts/<int:node_id>

Request type: GET

Response: Returns a JSON array of status, data and message.

Example response format:


```
{
        "status": "success",
        "message": "Node details is fetched successfully",
        "data": {
                "id": 1,
                "host_identifier": "EC2306BC-DCF7-A1F9-3ADE-CED9B00D49FB",
                "node_key": "6b38482b-2526-4b3a-bc1c-b5166dc7f57f",
                "last_ip": "13.234.136.159",
                "os_info": {
                        "name": "Ubuntu",
                        "build": "",
                        "major": "18",
                        "minor": "4",
                        "patch": "0",
                        "version": "18.04.2 LTS (Bionic Beaver)",
                        "codename": "bionic",
                        "platform": "ubuntu",
                        "platform_like": "debian"
                },
                "node_info": {
                        "computer_name": "ip-172-31-30-15",
                        "hardware_model": "HVM domU",
                        "hardware_serial": "ec2306bc-dcf7-a1f9-3ade-ced9b00d49fb",
                        "hardware_vendor": "Xen",
                        "physical_memory": "8362713088",
```

```
                "cpu_physical_cores": "2"
            },
            "network_info": [
                {
                  "mac": "02:4b:07:36:bd:fc",
                  "mask": "255.255.240.0",
                  "address": "172.31.30.15",
                  "enabled": "",
                  "description": "",
                  "manufacturer": "",
                  "connection_id": "",
                  "connection_status": ""
                }
            ],
            "last_checkin": "2020-06-24T05:02:59.956558",
            "enrolled_on": "2020-06-20T15:45:37.870494",
            "last_status": "2020-06-24T05:02:58.337353",
            "last_result": "2020-06-24T05:02:58.337353",
            "last_config": "2020-06-24T04:59:27.771166",
            "last_query_read": "2020-06-24T05:02:59.963197",
            "last_query_write": "2020-06-23T17:43:30.837109"
        }
}
```


View a host's alert distribution

Lists a host's alerts distribution by sources, rules.

URL: *https://<BASE_URL>/hosts/<string:host_identifier>/alerts/distribution*

https://<BASE_URL>/hosts/<int:node_id>/alerts/distribution

Request type: GET

Response: Returns a JSON array of status, data and message.

Example response format:


```
{
```

```
"status": "success",
"message": "Alerts distribution details are fetched for the host",
"data": {
        "sources": {
                "ioc": {
                        "INFO": 8,
                        "LOW": 0,
                        "WARNING": 199,
                        "CRITICAL": 0,
                        "TOTAL": 207
                },
                "rule": {
                        "INFO": 4,
                        "LOW": 0,
                        "WARNING": 0,
                        "CRITICAL": 2,
                        "TOTAL": 6
                },
                "virustotal": {
                        "INFO": 0,
                        "LOW": 0,
                        "WARNING": 0,
                        "CRITICAL": 0,
                        "TOTAL": 0
                },
                "ibmxforce": {
                        "INFO": 0,
                        "LOW": 0,
                        "WARNING": 0,
                        "CRITICAL": 0,
                        "TOTAL": 0
                },
```

```
                "alienvault": {
                        "INFO": 0,
                        "LOW": 0,
                        "WARNING": 0,
                        "CRITICAL": 0,
                        "TOTAL": 0
                }
        },
        "rules": [
                {
                        "name": "test_agg_rule1",
                        "count": 4
                },
                {
                        "name": "UAC Bypass via Event Viewer",
                        "count": 2
                }
        ]
    }
}
```


View hosts distribution count

Get count of hosts based on status and platform.

URL: *https://<BASE_URL>/hosts/count*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the nodes status count",
        "data": {
                "windows": {
```

```
                    "online": 0,
                    "offline": 2
                },
                "linux": {
                    "online": 0,
                    "offline": 1
                },
                "darwin": {
                    "online": 0,
                    "offline": 0
                }
            }
}
```


View status logs

Returns status logs of a host for the host identifier or node id given.

URL: *https://<BASE_URL>/hosts/status_logs*

Request type: POST

Example payload format:


```
{
        “host_identifier”: “EC2306BC-DCF7-A1F9-3ADE-CED9B00D49FB”,
        “node_id”:1,
        “start”:0,
        “limit”:10,
        “searchterm”:”EC2”
}
```


Required payload arguments: host_identifier / node_id

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the node's status logs",
```

```
        "data": {
                "results": [
                        {
                                "line": 922,
                                "message": "The chrome_extensions table returns data based on the current user by default, consider JOINing against the users table",
                                "severity": 1,
                                "filename": "virtual_table.cpp",
                                "created": "2020-06-21T00:59:32.768726",
                                "version": "4.0.2"
                        }
                ],
                "count": 197,
                "total_count": 197
        }
}
```


View additional config

Returns additional config of a host for the host identifier or node id given.

URL: *https://<BASE_URL>/hosts/additional_config*

Request type: POST

Example payload format:


```
{
        “host_identifier”: “EC2306BC-DCF7-A1F9-3ADE-CED9B00D49FB”,
        “node_id”:1
}
```


Required payload arguments: host_identifier / node_id

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
```

```
  "message": "Successfully fetched additional config of the node for the host identifier passed",
  "data": {
        "queries": [],
        "packs": [],
        "tags": [
            "test"
        ]
  }
}
```


View full config

Returns full config of a host for the host identifier or node id given.

URL: *https://<BASE_URL>/hosts/config*

Request type: POST

Example payload format:


```
{
    “host_identifier”: “EC2306BC-DCF7-A1F9-3ADE-CED9B00D49FB”,
    “node_id”:1
}
```


Required payload arguments: host_identifier / node_id

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
    "status": "success",
    "message": "Successfully fetched full config of the node for the host identifier passed",
    "data": {
      "options": {
            "disable_watchdog": true,
            "logger_tls_compress": true,
            "host_identifier": "uuid",
            "custom_plgx_enable_respserver": "true",
```

```
                "custom_plgx_EnableAgentRestart": "false"
        },
        "file_paths": {},
        "queries": {
                "win_process_events": {
                        "id": 125,
                        "query": "select * from win_process_events_optimized;",
                        "interval": 30,
                        "description": "Windows Process Events",
                        "status": true
                },
                "win_file_events": {
                        "id": 126,
                        "query": "select * from win_file_events_optimized;",
                        "interval": 180,
                        "description": "File Integrity Monitoring",
                        "status": true
                }
        },
        "packs": [],
        "filters": {}
  }
}
```


View count of result log

Returns result log count of a host for the host identifier or node id given.

URL: *https://<BASE_URL>/hosts/recent_activity/count*

Request type: POST

Example payload format:


```
{
        “host_identifier”: “EC2306BC-DCF7-A1F9-3ADE-CED9B00D49FB”,
        “node_id”:1
```

```
}
```


Required payload arguments: host_identifier / node_id

Response: Returns a JSON array data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the count of schedule query results count of host identifier passed",
        "data": [
                {
                        "name": "certificates",
                        "count": 514
                },
                {
                        "name": "drivers",
                        "count": 41
                }
        ]
}
```


View result log

Returns result log data of a host for a query for the host identifier or node id given.

URL: *https://<BASE_URL>/hosts/recent_activity*

Request type: POST

Example payload format:


```
{
        “host_identifier”: “EC2306BC-DCF7-A1F9-3ADE-CED9B00D49FB”,
        “node_id”:1,
        “query_name”:”certificates”,
        “start”:0,
        “limit”:2,
        “searchterm”:”EC2”,
```

```
        “column_name”: “md5”,
        “column_value”: “<Actual column value here>”
}
```


Required payload arguments: host_identifier / node_id, query_name, start and limit

Response: Returns a response json containing data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the schedule query results of host identifier passed",
        "data": {
                "count": 514,
                "total_count": 514,
                “categorized_count”: 310,
                "results": [
                        {
                                "timestamp": "06/20/2020 18/05/15",
                                "action": "added",
                                "columns": {
                                  "path": "LocalMachine\\Windows Live ID Token Issuer",
                                  "issuer": "Token Signing Public Key",
                                  "common_name": "Token Signing Public Key",
                                  "self_signed": "1",
                                  "not_valid_after": "1530479437"
                                }
                        },
                        {
                                "timestamp": "06/20/2020 18/05/15",
                                "action": "added",
                                "columns": {
                                  "path": "LocalMachine\\Windows Live ID Token Issuer",
                                  "issuer": "Token Signing Public Key",
```

```
                                    "common_name": "Token Signing Public Key",
                                    "self_signed": "1",
                                    "not_valid_after": "1620506455"
                                }
                            }
                        ]
                }
}
```


View list of tags of a host

Returns list of tags of a host for the host identifier or node id given.

URL: *https://<BASE_URL>/hosts/<string:host_identifier>/tags*

https://<BASE_URL>/hosts/<int:node_id>/tags

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the tags of host",
        "data": [
                "test"
        ]
}
```


Create tags to a host

Creates tags to a host.

URL: *https://<BASE_URL>/hosts/<string:host_identifier>/tags*

https://<BASE_URL>/hosts/<int:node_id>/tags

Request type: POST

Example payload format:


```
{
```

```
    "tag": "test"
}
```

Required payload arguments: tag

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
    "status": "success",
    "message": "Successfully created tags to host"
}
```

Remove tags from a host

Remove tags of a host for the host identifier given.

URL: *https://<BASE_URL>/hosts/<string:host_identifier>/tags*

https://<BASE_URL>/hosts/<int:node_id>/tags

Request type: DELETE

Example payload format:

```
{
    "tag": "simple"
}
```

Required payload arguments: tag

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
    "status": "success",
    "message": "Successfully removed tags from host"
}
```

Search export

Exports the search results of a host into csv file.

URL: *https://<BASE_URL>/hosts/search/export*

Request type: POST

Example payload format:

```
{
        "conditions": {
                "condition": "OR",
                "rules": [
                        {
                                "id": "name",
                                "field": "name",
                                "type": "string",
                                "input": "text",
                                "operator": "contains",
                                "value": "EC2"
                        },
                        {
                                "id": "name",
                                "field": "name",
                                "type": "string",
                                "input": "text",
                                "operator": "equal",
                                "value": "pc"
                        }
                ],
                "valid": true
        },
        “host_identifier”:”EC241E83-BDC2-CAFC-BF9F-28C22B37A7F0”,
        “query_name”:”win_file_events”,
        “column_name”: “md5”,
        “column_value”: “<Actual column value here>”
}
```

Required payload arguments: conditions, host_identifier and query_name.

Response: Returns a CSV file.

Note: We maintain 3 states for the enrolled hosts in EclecticIQ ESP platform.

1. *Enabled – Host will be allowed for all activity and will be shown in every page except Resolved hosts page in management section.*
2. *Removed – Host will be restricted from any kind of activity from agent and will be hidden everywhere except Resolved hosts page in management section. Re-enrollment will be completely restricted till we re-enable it to use further.*
3. *Deleted – Host will be restricted from all activity and will be deleted from database too when the next purge cycle is started. Re-enrolling agent creates a new entry in platform user interface.*

Delete a host permanently

Delete a host permanently for the host identifier or node id given. It moved the host to stage-3 like mentioned above.

URL: *https://<BASE_URL>/hosts/<string:host_identifier>/delete*

https://<BASE_URL>/hosts/<int:node_id>/delete

Request type: DELETE

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully deleted the host"
}
```

Remove a host

Remove a host from the platform for the host identifier or node id given. It moved the host to stage-2 like mentioned above.

URL: *https://<BASE_URL>/hosts/<string:host_identifier>/delete*

https://<BASE_URL>/hosts/<int:node_id>/delete

Request type: PUT

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully removed the host"
}
```

Enable a host

Enable a host for the host identifier or node id given. It moved the host to stage-1 like mentioned above to allow for all activity.

URL: *https://<BASE_URL>/hosts/<string:host_identifier>/enable*

https://<BASE_URL>/hosts/<int:node_id>/enable

Request type: PUT

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully enabled the host"
}
```

Status log export.

Exports the Status log of a host into csv file.

URL: *https://<BASE_URL>/hosts/status_log/export*

Request type: POST

Example payload format:

```
{
        "host_identifier": "EC2EBD16-8D48-0C24-EB42-B22333D7F08D",
        "node_id": 8,
        "searchterm":"EC2"
}
```

Required payload arguments: host_identifier / node_id.

Response:

```
{
  "status": "Success",
 "message": "Downloading will be completed in sometime",
 "data": {
   "task_id": "929e5634-2d21-4f3a-923d-659916365670"
 }
}
```

--> Make a connection from a socketio client to the below URL.

wss://<IP_OF_THE_SERVER>/websocket/csv/export

--> Emit below payload to the socket server.

{“task_id”: <task_id _from_api_response>}

For ex: *{“task_id": "929e5634-2d21-4f3a-923d-659916365670"}*

--> Keep emitting the message ‘pong’ when ‘ping’ is received.

--> Keep the socket client listen to server till download path is received.

BLUEPRINT: tags

Blueprint-path: /tags

View list of all tags

Returns list of all tags.

URL: *https://<BASE_URL>/tags*

Request type: GET

Example payload format:

```
{
        “searchterm”:“test”,
        “start”:0,
        “limit”:10
}
```

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
         "status": "success",
         "message": "Successfully fetched the tags info",
         "data": {
                  "count": 7,
                  "total_count": 7,
                  "results": [
                           {
```

```
                "value": "test67",
                "nodes": [],
                "packs": [],
                "queries": [],
                "file_paths": []
            },
            {
                "value": "test",
                "nodes": [],
                "packs": [
                  "all-events-pack"
                ],
                "queries": [
                  "App_disabledExceptionChainValidation"
                ],
                "file_paths": []
            }
        ]
    }
}
```


Add a tag

Adds a tag.

URL: *https://<BASE_URL>/tags/add*

Request type: POST

Example payload format:


```
{
    “tag": "test”
}
```


Required payload arguments: tag

Response: Returns a JSON array of status and message.

Example response format:

```
{
        “status”: “success”,
        “message”: “Tag is added successfully”,
}
```

Delete a tag

Deletes a tag.

URL: *https://<BASE_URL>/tags/delete*

Request type: POST

Example payload format:

```
{
        “tag": "test”
}
```

Required payload arguments: tag

Response: Returns a JSON array of status and message.

Example response format:

```
{
        “status”: “success”,
        “message”: “Tag is deleted successfully”,
}
```

View all hosts, packs, queries of a tag

Get list of all hosts, packs and queries of a tag.

URL: *https://<BASE_URL>/tags/tagged*

Request type: POST

Example payload format:

```
{
        “tags": "test”
}
```

Required payload arguments: tags

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "All hosts, queries, packs for the tag provided!",
        "data": {
                "hosts": [
                        {
                                "id": 3,
                                "display_name": "EC2AMAZ-2RJ1BIF",
                                "host_identifier": "EC2CE2E2-3D74-1248-2FA9-23F2E960ED42",
                                "os_info": {
                                 "name": "windows"
                                },
                                "tags": [
                                 "test"
                                ],
                                "last_ip": "15.206.168.222",
                                "is_active": false
                        }
                ],
                "packs": [
                        {
                                "id": 1,
                                "name": "all-events-pack",
                                "platform": null,
                                "version": null,
                                "description": null,
                                "shard": null,
                                "category": "General",
                                "tags": [
                                 "test"
                                ],
```

```
                "queries": [
                 {
                        "id": 2,
                        "name": "win_process_events",
                        "sql": "select * from win_process_events;",
                        "interval": 38,
                        "platform": "windows",
                        "version": "2.9.0",
                        "description": "Windows Process Events",
                        "value": "Process Events",
                        "snapshot": false,
                        "shard": null,
                        "tags": [],
                        "packs": [
                          "all-events-pack"
                        ]
                    }
                    ],
"queries": [
 {
            "id": 2,
            "name": "win_process_events",
            "sql": "select * from win_process_events;",
            "interval": 38,
            "platform": "windows",
            "version": "2.9.0",
            "description": "Windows Process Events",
            "value": "Process Events",
            "snapshot": false,
            "shard": null,
            "tags": [],
            "packs": [
```

```
                    "all-events-pack"
                ]
            }
            ]
        }
        ]
    }
}
```


BLUEPRINT: carves

Blueprint-path: /carves

View all carves

Lists all carves.

URL: *https://<BASE_URL>/carves*

Request type: POST

Example payload format:


```
{
    "host_identifier":"77858CB1-6C24-584F-A28A-E054093C8924",
    “start”:0,
    “limit”:10
}
```


Filters description:

host_identifier – pass value to this argument to filter the records by a host

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
    "status": "success",
    "message": "Successfully fetched the Carves data",
    "data": {
        "count": 1,
        "results": [
            {
```

```
                "id": 1,
                "node_id": 2,
                "session_id": "793OF12PEQ",
                "carve_guid": "3ecdb82c-5d6f-4c0f-b532-bdcb2588894d",
                "carve_size": 34766848,
                "block_size": 300000,
                "block_count": 116,
                "archive": "793OF12PEQ3ecdb82c-5d6f-4c0f-b532-
            bdcb2588894d.tar",
                "status": "COMPLETED",
                "created_at": "2020-06-29T10:41:41.532733",
                "hostname": "EC2AMAZ-5FTJV7B"
            }
        ]
    }
}
```


Download a carve

Returns a file object of Carves.

URL: *https://<BASE_URL>/carves/download/<string:session_id>*

Request type: GET

Response: Returns a file.

Carve through query and host identifier

Get the Carve details for the distributed query id and host identifier given.

URL: *https://<BASE_URL>/carves/query*

Request type: POST

Example payload format:


```
{
    "host_identifier":"77858CB1-6C24-584F-A28A-E054093C8924",
    “query_id”:10
}
```

Filters description:

host_identifier – pass value to this argument to filter the records by a host

query_id – distributed query id used to make carve

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
    "status": "success",
    "message": "Successfully fetched the Carve",
    "data": {
        "count": 1,
        "results": [
            {
                "id": 1,
                "node_id": 2,
                "session_id": "793OF12PEQ",
                "carve_guid": "3ecdb82c-5d6f-4c0f-b532-bdcb2588894d",
                "carve_size": 34766848,
                "block_size": 300000,
                "block_count": 116,
                "archive": "793OF12PEQ3ecdb82c-5d6f-4c0f-b532-
                bdcb2588894d.tar",
                "status": "COMPLETED",
                "created_at": "2020-06-29T10:41:41.532733",
                "hostname": "EC2AMAZ-5FTJV7B"
            }
        ]
    }
}
```


Delete a carve

Deletes a carve.

URL: *https://<BASE_URL>/carves/delete*

Request type: POST

Example response format:

```
{
        “session_id”:” 793OF12PEQ”
}
```

Required payload arguments: session_id

Response: Returns a JSON array of status and message.

Example response format:

```
{
        “status”:”success”,
        “message”:”Carve is deleted successfully”
}
```

BLUEPRINT: distributed

Blueprint-path: /distributed

Add distributed (live) queries

Adds distributed queries.

URL: *https://<BASE_URL>/distributed/add*

Request type: POST

Example payload format:

```
{
        "tags": "demo",
        "query": "select * from system_info;”,
        "nodes": "6357CE4F-5C62-4F4C-B2D6-CAC567BD6113,6357CE4F-5C62-4F4C-B2D6-
        CAGF12F17F23”,
        “description”:”live query to get system_info”
}
```

Required payload arguments: query, nodes/tags

Response: Returns a JSON array of query_id, status and message.

Example response format:

```
{
         "status": "success",
```

```
    "message": "Distributed query is sent successfully",
    "data": {
        "query_id": 200,
        "onlineNodes": 3
    }
}
```

--> Make a connection from a socketio client to the below URL.

wss://<IP_OF_THE_SERVER>/distributed/result

--> Emit below payload to the socket server.

{“query_id”:<query_id_from_api_response>}

For ex: *{“query_id”:2}*

--> Keep emitting the message ‘pong’ when ‘ping’ is received.

--> Keep the socket client listen to server till a message with format is received.

```
{
    "node": {
        "id": 6,
        "name": "ip-172-31-16-229"
    },
    "data": [
        {
            "uid": "0",
            "gid": "0",
            "uid_signed": "0",
            "gid_signed": "0",
            "username": "root",
            "description": "root",
            "directory": "/root",
            "shell": "/bin/bash",
            "uuid": ""
        }
    ],
```

```
        "query_id": 2
    }
```

BLUEPRINT: yara

Blueprint-path: /yara

View YARA files list

Returns list of yara file names.

URL: *https://<BASE_URL>/yara*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
    "status": "success",
    "message": "Successfully fetched the yara files “,
    “data”: [“data.txt”,” sample.txt”]
}
```

Upload YARA file

Uploads a yara file to the EclecticIQ ESP.

URL: *https://<BASE_URL>/yara/add*

Request type: POST

Example payload format:

```
{
    “file”: <An Yara file>
}
```

Required payload arguments: file

Response: Returns a JSON array of status and message.

Example response format:

```
{
     "status": "success",
     "message": "Successfully uploaded the file"
```

}

View content of YARA file

Returns the content of the yara file.

URL: *https://<BASE_URL>/yara/view*

Request type: POST

Example payload format:

```
{
        "file_name":"eicar.yara"
}
```

Required payload arguments: file_name

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the yara file content!",
        "data": "rule eicar_av_test {\n    /*\n      Per standard, match only if entire file is EICAR string plus optional trailing whitespace.\n      The raw EICAR string to be matched is:\n X5O!P%@AP[4\\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*\n */\n\n    meta:\n      description = \"This is a standard AV test, intended to verify that BinaryAlert is working correctly.\"\n      author = \"Austin Byers | Airbnb CSIRT\"\n reference = \"http://www.eicar.org/86-0-Intended-use.html\"\n\n    strings:\n $eicar_regex = /^X5O!P%@AP\\[4\\\\PZX54\\(P\\^\\)7CC\\)7\\}\\$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!\\$H\\+H\\*\\s*$/\n\n    condition:\n      all of them\n}\n\nrule eicar_substring_test {\n    /*\n      More generic - match just the embedded EICAR string (e.g. in packed executables, PDFs, etc)\n    */\n\n    meta:\n      description = \"Standard AV test, checking for an EICAR substring\"\n      author = \"Austin Byers | Airbnb CSIRT\"\n\n strings:\n      $eicar_substring = \"$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!\"\n\n condition:\n      all of them\n}"
}
```


Delete a YARA file

Deletes a yara file for the name given.

URL: *https://<BASE_URL>/yara/delete*

Request type: POST

Example payload format:

```
{
```

```
        “file_name”:”eicar.yara”
}
```

Required payload arguments: file_name

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": "File with the given file name is deleted successfully"
}
```

BLUEPRINT: iocs

Blueprint-path: /iocs

View IOCs

Returns existing IOCs.

URL: *https://<BASE_URL>/iocs*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully fetched the iocs",
        "data": {
                "test-intel_ipv4": {
                        "type": "remote_address",
                        "severity": "WARNING",
                        "intel_type": "self",
                        "values": "3.30.1.15,3.30.1.16"
                },
                "test-intel_domain_name": {
                        "type": "domain_name",
                        "severity": "WARNING",
```

```
                "intel_type": "self",
                "values": "unknown.com,slackabc.com"
            },
            "test-intel_md5": {
                "type": "md5",
                "severity": "INFO",
                "intel_type": "self",
                "values": "3h8dk0sksm0,9sd772ndd80"
            }
        }
}
```


Update IOCs

Update iocs json.

URL: *https://<BASE_URL>/iocs/add*

Request type: POST

Example payload format:


```
{
    "data": {
            "test-intel_ipv4": {
                "type": "remote_address",
                "severity": "WARNING",
                "intel_type": "self",
                "values": "3.30.1.15,3.30.1.16"
            },
            "test-intel_domain_name": {
                "type": "domain_name",
                "severity": "WARNING",
                "intel_type": "self",
                "values": "unknown.com,slackabc.com"
            },
            "test-intel_md5": {
```

```
                    "type": "md5",
                    "severity": "INFO",
                    "intel_type": "self",
                    "values": "3h8dk0sksm0,9sd772ndd80"
                }
            }
}
```


Required payload arguments: data

Response: Returns a JSON array of status and message.

Example response format:


```
{
    "status": "success",
    "message": "Successfully updated the intel data “
}
```


BLUEPRINT: email

Blueprint-path: /email

Configure email settings

Configures email data like recipients, sender, smtp port.

URL: *https://<BASE_URL>/email/configure*

Request type: POST

Example payload format:


```
{
    "emailRecipients": " eclecticiqsample@gmail.com,mousegame@gmail.com",
    "email": "mousegame@gmail.com",
    "smtpAddress": "smtp2.gmail.com",
    "password": "a",
    "smtpPort": 445 ,
    “use_ssl”: false,
    “use_tls”: false
}
```

Required payload arguments: emailRecipients, email, smtpAddress, password and smtpPort

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully updated the email configuration “,
        “data”: {
                "emailRecipients": [" eclecticiqsample@gmail.com","mousegame@gmail.com"],
                "email": "mousegame@gmail.com",
                "smtpAddress": "smtp2.gmail.com",
                "password": "YQ==\n",
                "smtpPort": 445 ,
                “use_ssl”: false,
                “use_tls”: false
        }
}
```


View email settings

Returns existing email data like recipients, sender, smtp port.

URL: *https://<BASE_URL>/email/configure*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the email configuration “,
        “data”: {
                "emailRecipients": [" eclecticiqsample@gmail.com","mousegame@gmail.com"],
                "email": "mousegame@gmail.com",
                "smtpAddress": "smtp2.gmail.com",
                "password": "YQ==\n",
                "smtpPort": 445,
```

```
        "use_ssl": false,
        "use_tls": false
    }
}
```

Test email

Sends an email and validates the config is valid or not.

URL: *https://<BASE_URL>/email/test*

Request type: POST

Example payload format:

```
{
    "emailRecipients": " eclecticiqsample@gmail.com,mousegame@gmail.com",
    "email": "mousegame@gmail.com",
    "smtpAddress": "smtp2.gmail.com",
    "password": "a",
    "smtpPort": 445 ,
    "use_ssl": false,
    "use_tls": false
}
```

Required payload arguments: emailRecipients, email, smtpAddress, password and smtpPort

Response: Returns a JSON array of status and message.

Example response format:

```
{
    "status": "success",
    "message": "A Test mail is sent to the recipients successfully "
}
```

BLUEPRINT: schema

Blueprint-path: /schema

View OSQuery schema

Returns all OSQuery tables schema.

URL: *https://<BASE_URL>/schema*

Request type: GET

Example payload format:

```
{
        “export_type”:"json”
}
```

Filters description:

export_type – json/sql – json to get the schema in json format, sql to get the schema in SQL queries.

Response: Returns a JSON array of data, status and message.

Example response format: sql format


```
{
        "status": "success",
        "message": "Successfully fetched the schema",
        "data": {
                "account_policy_data": "CREATE TABLE account_policy_data (uid BIGINT,
                creation_time DOUBLE, failed_login_count BIGINT, failed_login_timestamp DOUBLE,
                password_last_set_time DOUBLE)",
                "acpi_tables": "CREATE TABLE acpi_tables (name TEXT, size INTEGER, md5 TEXT)",
        }
}
```


Example response format: json format


```
{
        "status": "success",
        "message": " EclecticIQ ESP agent schema is fetched successfully",
        "data": [
                {
                        "name": "etc_hosts",
                        "description": "Line-parsed /etc/hosts.",
                        "platform": [
                                "windows",
                                "linux",
                                "darwin",
```

```
                "freebsd",
                "posix"
            ],
            "schema": {
                "address": {
                  "type": "TEXT",
                  "description": "IP address mapping",
                  "is_required": false
                },
                "hostnames": {
                  "type": "TEXT",
                  "description": "Raw hosts mapping",
                  "is_required": false
                }
            }
        }
    }
```


View one OSQuery table's schema

Returns an OSQuery table schema for the table name given.

URL: *https://<BASE_URL>/schema/<string:table>*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
    "status": "success",
    "message": "Successfully fetched the table schema",
    "data": {
        "account_policy_data": "CREATE TABLE account_policy_data (uid BIGINT,
        creation_time DOUBLE, failed_login_count BIGINT, failed_login_timestamp DOUBLE,
        password_last_set_time DOUBLE)"
    }
}
```

BLUEPRINT: rules

Blueprint-path: /rules

View all rules

Returns all rules.

URL: *https://<BASE_URL>/rules*

Request type: POST

Example payload format:

```
{
        “start”:0,
        “limit”:1,
        “searchterm”:”EC”,
        “alerts_count”:true
}
```

Response: Returns a JSON array of data, status and message.

Filters description:

alerts_count – true/false – true to get non resolved alerts count of the rule

Example response format:

```
{
        "status": "success",
        "message": "Successfully fetched the rules info",
        "data": {
                "count": 147,
                "total_count": 147,
                "results": [
                        {
                          "id": 147,
                          "alerters": [
                            "debug"
                          ],
```

```
                "conditions": {
                 "rules": [
                        {
                                "id": "action",
                                "type": "string",
                                "field": "action",
                                "input": "text",
                                "value": "test",
                                "operator": "equal"
                        }
                 ],
                 "valid": true,
                 "condition": "AND"
                },
                "description": "tesing",
                "name": "test123",
                "severity": "INFO",
                "status": "ACTIVE",
                "updated_at": "2020-06-30T07:46:00.265400",
                "type": "MITRE",
                "tactics": [
                 "defense-evasion"
                ],
                "technique_id": "T1070",
                "alerts_count": 23
               }
        ]
    }
}
```


View a rule

Returns a rule info for the id given.

URL: *https://<BASE_URL>/rules/<int:rule_id>*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the rules info",
        "data": {
                "id": 147,
                "alerters": [
                 "debug"
                ],
                "conditions": {
                 "rules": [
                        {
                                "id": "action",
                                "type": "string",
                                "field": "action",
                                "input": "text",
                                "value": "test",
                                "operator": "equal"
                        }
                 ],
                 "valid": true,
                 "condition": "AND"
                },
                "description": "tesing",
                "name": "test123",
                "severity": "INFO",
                "status": "ACTIVE",
                "updated_at": "2020-06-30T07:46:00.265400",
```

```
                    "type": "MITRE",
                    "tactics": [
                      "defense-evasion"
                    ],
                    "technique_id": "T1070"
            }
}
```


Modify a rule

Edits and Returns a rule info for the id, data given.

URL: *https://<BASE_URL>/rules/<int:rule_id>*

Request type: POST

Example payload format:


```
{
    "alerters": "debug,email",
    "conditions": {
      "rules": [
                {
                        "id": "action",
                        "type": "string",
                        "field": "action",
                        "input": "text",
                        "value": "test",
                        "operator": "equal"
                }
      ],
      "valid": true,
      "condition": "AND"
    },
    "description": "tesing",
    "name": "test123",
    "severity": "INFO",
```

```
    "status": "ACTIVE",
    "updated_at": "2020-06-30T07:46:00.265400",
    "type": "MITRE",
    "tactics": "defense-evasion",
    "technique_id": "T1070, T1005"
}
```


Required payload arguments: name and conditions

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully modified the rules info",
        "data": {
                "id": 147,
                "alerters": [
                 "debug"
                ],
                "conditions": {
                 "rules": [
                        {
                                "id": "action",
                                "type": "string",
                                "field": "action",
                                "input": "text",
                                "value": "test",
                                "operator": "equal"
                        }
                 ],
                 "valid": true,
                 "condition": "AND"
                },
                "description": "tesing",
```

```
                    "name": "test123",
                    "severity": "INFO",
                    "status": "ACTIVE",
                    "updated_at": "2020-06-30T07:46:00.265400",
                    "type": "MITRE",
                    "tactics": [
                      "defense-evasion"
                    ],
                    "technique_id": "T1070"
            }
}
```


Add a rule

Adds a rule for the data given.

URL: *https://<BASE_URL>/rules/add*

Request type: POST

Example payload format:


```
{
    "alerters": "debug,email",
    "conditions": {
      "rules": [
                {
                    "id": "action",
                    "type": "string",
                    "field": "action",
                    "input": "text",
                    "value": "test",
                    "operator": "equal"
                }
      ],
      "valid": true,
      "condition": "AND"
```

```
    },
    "description": "tesing",
    "name": "test123",
    "severity": "INFO",
    "status": "ACTIVE",
    "updated_at": "2020-06-30T07:46:00.265400",
    "type": "MITRE",
    "tactics": "defense-evasion",
    "technique_id": "T1070, T1005"
}
```


Required payload arguments: name and conditions

Response: Returns a JSON array of rule_id, status and message.

Example response format:


```
{
        "status": "success",
        "message": “Rule is added successfully “,
        “rule_id”: 2
}
```


Get tactics for technique ids

Returns tactics for the technique ids given.

URL: *https://<BASE_URL>/rules/tactics*

Request type: POST

Example payload format:


```
{
        “technique_ids”:” T1005, T1004”
}
```


Required payload arguments: technique_ids

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
         "status": "success",
```

```
    "message": "Tactics are fetched successfully from technique ids",
    "data": {
        "tactics": [
          "collection"
        ],
        "description": "\nSensitive data can be collected from local system sources, such as the file system or databases of information residing on the system prior to Exfiltration.\n\nAdversaries will often search the file system on computers they have compromised to find files of interest. They may do this using a [Command-Line Interface](https://attack.mitre.org/techniques/T1059), such as [cmd](https://attack.mitre.org/software/S0106), which has functionality to interact with the file system to gather information. Some adversaries may also use [Automated Collection](https://attack.mitre.org/techniques/T1119) on the local system.\n"
    }
}
```


BLUEPRINT: queries

Blueprint-path: /queries

View all queries

Returns all queries.

URL: *https://<BASE_URL>/queries*

Request type: POST

Example payload format:


```
{
    "start":0,
    "limit":1,
    “searchterm”:”EC2”
}
```


Response: Returns a JSON array of data, status and message.

Example response format:


```
{
    "status": "success",
```

```
        "message": "Successfully fetched the queries info!",
        "data": {
                "count": 103,
                "total_count": 103,
                "results": [
                        {
                                "id": 78,
                                "name": "AppCompat",
                                "sql": "select * from registry where
                            key='HKEY_LOCAL_MACHINE\\SOFTWARE\\%Microsoft\\Windows
                            NT\\CurrentVersion\\AppCompatFlags\\Layers'",
                                "interval": 86400,
                                "platform": null,
                                "version": null,
                                "description": "Check Applications opted in for DEP",
                                "value": null,
                                "snapshot": true,
                                "shard": null,
                                "tags": [],
                                "packs": [
                                  "windows-hardening"
                                ]
                        }
                ]
        }
}
```


View all packed queries

Returns all packed queries.

URL: *https://<BASE_URL>/queries/packed*

Request type: POST

Example payload format:

```
{
        "start":0,
        "limit":1,
        “searchterm”:””
}
```

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully fetched the packed queries info",
        "data": {
                "count": 1,
                “total_count”:103,
                "results": [
                        {
                                "id": 78,
                                "name": "AppCompat",
                                "sql": "select * from registry where
                            key='HKEY_LOCAL_MACHINE\\SOFTWARE\\%Microsoft\\Windows
                            NT\\CurrentVersion\\AppCompatFlags\\Layers'",
                                "interval": 86400,
                                "platform": null,
                                "version": null,
                                "description": "Check Applications opted in for DEP",
                                "value": null,
                                "snapshot": true,
                                "shard": null,
                                "tags": [],
                                "packs": [
                                  "windows-hardening"
                                ]
                        }
                ]
```

```
        }
}
```


View a query

Returns a query info for the id given.

URL: *https://<BASE_URL>/queries/<int:query_id>*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the query info for the given id",
        "data": {
                "id": 78,
                "name": "AppCompat",
                "sql": "select * from registry where
            key='HKEY_LOCAL_MACHINE\\SOFTWARE\\%Microsoft\\Windows
            NT\\CurrentVersion\\AppCompatFlags\\Layers'",
                "interval": 86400,
                "platform": null,
                "version": null,
                "description": "Check Applications opted in for DEP",
                "value": null,
                "snapshot": true,
                "shard": null,
                "tags": [],
                "packs": [
                  "windows-hardening"
                ]
        }
}
```


Add a query

Adds a query for the data given.

URL: *https://<BASE_URL>/queries/add*

Request type: POST

Example payload format:

```
{
        "name": "running_process_query",
        "query": "select * from processes;",
        "interval": 5,
        "platform": "windows",
        "version": "2.9.0",
        "snapshot": "true",
        "description": "Processes",
        "value": "Processes",
        "tags":"finance,sales"
}
```

Required payload arguments: name, query and interval

Response: Returns a JSON array of query_id, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully added the query for the data given",
        "query_id": 2
}
```

Modify a query

Edits a query for the id given.

URL: *https://<BASE_URL>/queries/<int:query_id>*

Request type: POST

Example payload format:

```
{
        "name": "running_process_query",
        "query": "select * from processes;",
```

```
        "interval": 5,
        "platform": "windows",
        "version": "2.9.0",
        "snapshot": "true",
        "description": "Processes",
        "value": "Processes”,
        "tags":"finance,sales"
}
```

Required payload arguments: name, query and interval

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully edited the query info for the given id",
        "data": {
                "id": 78,
                "name": "AppCompat",
                "sql": "select * from registry where
            key='HKEY_LOCAL_MACHINE\\SOFTWARE\\%Microsoft\\Windows
            NT\\CurrentVersion\\AppCompatFlags\\Layers'",
                "interval": 86400,
                "platform": "all",
                "version": null,
                "description": "Check Applications opted in for DEP",
                "value": null,
                "snapshot": true,
                "shard": null
        }
}
```

View tags of a query

Modifies tags for a query for id given.

URL: *https://<BASE_URL>/queries/<int:query_id/tags*

Request type: GET

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully fetched the tags of query",
        "data": [
          "test"
        ]
}
```

Add tags to a query

Adds tags to a query for id given.

URL: *https://<BASE_URL>/queries/<int:query_id>/tags*

Request type: POST

Example payload format:

```
{
        “tag":"finance"
}
```

Required payload arguments: tag

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": “Successfully created the tag(s) to queries”
}
```

Delete tags from a query

Removes tags of a query for id given.

URL: *https://<BASE_URL>/queries/<int:query_id>/tags*

Request type: DELETE

Example payload format:

```
{
        “tag":"finance"
}
```

Required payload arguments: tag

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully removed tags from query"
}
```

Delete a query

Delete a query for id given.

URL: *https://<BASE_URL>/queries/<int:query_id>/delete*

https://<BASE_URL>/queries/<string:query_name>/delete

Request type: DELETE

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": “Successfully deleted the query”
}
```

BLUEPRINT: packs

Blueprint-path: /packs

View all packs

Returns all Packs.

URL: *https://<BASE_URL>/packs*

Request type: POST

Example payload format:

```
{
```

```
“start”:0,
“limit”:1,
“searchterm”:””
}
```


Response: Returns a JSON array of data, status and message.

Example response format:


```
{
 "status": "success",
 "message": "successfully fetched the packs info",
 "data": {
  "count": 1,
  “total_count”:12,
  "results": [
   {
    "id": 12,
    "name": "windows-hardening",
    "platform": null,
    "version": null,
    "description": null,
    "shard": null,
    "category": "General",
    "tags": [],
    "queries": [
     {
      "id": 82,
      "name": "PolicyScopeMachine",
      "sql": "select * from registry where
key='HKEY_LOCAL_MACHINE\\SOFTWARE\\Policies\\Micros
oft\\Windows\\Safer\\CodeIdentifiers\\PolicyScope'",
      "interval": 86400,
      "platform": null,
      "version": null,
```

```
                    "description": "Check Software Restriction Policies state",
                    "value": null,
                    "snapshot": true,
                    "shard": null,
                    "tags": [],
                    "packs": [
                     "windows-hardening"
                    ]
                }
            ]
        }
    ]
}
```


View a pack

Returns a pack for the id given.

URL: *https://<BASE_URL>/packs/<int:pack_id>*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
    "status": "success",
    "message": "successfully fetched the packs info",
    "data": {
            "id": 12,
            "name": "windows-hardening",
            "platform": null,
            "version": null,
            "description": null,
            "shard": null,
```

```
                "category": "General",
                "tags": [],
                "queries": [
                        {
                                "id": 82,
                                "name": "PolicyScopeMachine",
                                "sql": "select * from registry where
                            key='HKEY_LOCAL_MACHINE\\SOFTWARE\\Policies\\Microsoft\\Win
                            dows\\Safer\\CodeIdentifiers\\PolicyScope'",
                                "interval": 86400,
                                "platform": null,
                                "version": null,
                                "description": "Check Software Restriction Policies state",
                                "value": null,
                                "snapshot": true,
                                "shard": null,
                                "tags": [],
                                "packs": [
                                  "windows-hardening"
                                ]
                        }
                ]
        }
}
```


Add a pack

Adds a pack for the data given.

URL: *https://<BASE_URL>/packs/add*

Request type: POST

Example payload format:


```
{
        "name": "process_query_pack",
```

```
        "queries": {
                "win_file_events": {
                        "query": "select * from processes;",
                        "interval": 5,
                        "platform": "windows",
                        "version": "2.9.0",
                        "description": "Processes",
                        "value": "Processes"
                }
        },
        "tags": "finance, sales",
        “category": "General”
}
```

Required payload arguments: name, queries

Response: Returns a JSON array of pack_id, status and message.

Example response format:

```
{
        “status": "success”,
        “message": "Imported query pack and pack is added successfully”,
        “pack_id”:2
}
```

View tags of a pack

Lists tags for a pack for id given.

URL: *https://<BASE_URL>/packs/<int:pack_id/tags*

https://<BASE_URL>/packs/<string:pack_name>/tags

Request type: GET

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully fetched the tags of pack",
```

```
    "data": [
      "test"
    ]
}
```

Add tags to a pack

Adds tags to a pack for id given.

URL: *https://<BASE_URL>/packs/<int:pack_id>/tags*

https://<BASE_URL>/packs/<string:pack_name>/tags

Request type: POST

Example payload format:

```
{
    “tag": "finance"
}
```

Required payload arguments: tag

Response: Returns a JSON array of status and message.

Example response format:

```
{
    "status": "success",
    "message": “Successfully created the tag(s) to packs”
}
```

Delete tags from a pack

Removes tags of a pack for id given.

URL: *https://<BASE_URL>/packs/<int:pack_id>/tags*

https://<BASE_URL>/packs/<string:pack_name>/tags

Request type: DELETE

Example payload format:

```
{
    “tag": "finance"
}
```

Required payload arguments: tag

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully removed tags from pack"
}
```

Delete a pack

Delete a pack for id given.

URL: *https://<BASE_URL>/packs/<int:pack_id>/delete*

https://<BASE_URL>/packs/<string:pack_name>/delete

Request type: DELETE

Response: Returns a JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": “Successfully deleted the pack”
}
```

Upload a pack

Adds pack through a file upload.

URL: *https://<BASE_URL>/packs/upload*

Request type: POST

Example payload format:

```
{
        “file”: <A JSON file with json content same as /packs/add but without pack name>,
        “category”: “General”
}
```

Required payload arguments: file and category

Response: Returns a JSON array of pack id, status and message.

Example response format:

```
{
```

```
    “status": "success”,
    “message": "pack uploaded successfully”,
    “pack_id”:2
}
```

BLUEPRINT: configs

Blueprint-path: /configs

Add a new config from another config.

Adds a new config by copying other config’s queries and filters.

URL: *https://<BASE_URL>/configs*

Request type: POST

Example payload format:

```
{
    "name":"test",
    "platform": "linux",
    "queries": {
        "process_events": {
            "interval": 10,
            "status": true
            },
        "osquery_info": {
            "interval": 86400,
            "status": true
            }
        },
    "filters": {},
    "config_id":1
}
```

Required payload arguments: platform, queries, filters and name

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Config is added successfully",
        "data": 6
}
```


Returns all configs.

Returns all configs present in db.

URL: *https://<BASE_URL>/configs*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the config data",
        "data": {
                "linux": {
                        "auto-conf-linux": {
                                "queries": {
                                        "arp_cache": {
                                                "id": 300,
                                                "query": "select * from arp_cache;",
                                                "interval": 86400,
                                                "platform": "linux",
                                                "snapshot": false,
                                                "status": true
                                        }
                                },
                                "filters": {
                                        "events": {
                                                "disable_subscribers": [
```

```
                                "user_events"
                        ]
                    },
                    "options": {
                        "custom_plgx_LogLevel": "1"
                    },
                    "file_paths": {
                        "binaries": [
                            "/usr/bin/%%",
                            "/usr/sbin/%%"
                        ],
                        "configuration": [
                            "/etc/passwd",
                            "/etc/shadow"
                        ]
                    }
                },
                "is_default": false,
                "id": 8,
                "conditions": {
                    "os_name": {
                        "value": "*Ubuntu*"
                    },
                    "hostname": {
                        "value": "*ip*"
                    }
                },
                "description": "auto"
            }
        }
}
```

```
}
```


Returns config for the ID given.

Returns full dict config for the ID given.

URL: *https://<BASE_URL>/configs /<int:config_id>*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the config data",
        "data": {
                “name": “auto-config-linux",
                "queries": {
                        "arp_cache": {
                                "id": 300,
                                "query": "select * from arp_cache;",
                                "interval": 86400,
                                "platform": "linux",
                                "snapshot": false,
                                "status": true
                        }
                },
                "filters": {
                        "events": {
                                "disable_subscribers": [
                                        "user_events"
                                ]
                        },
                        "options": {
                                "custom_plgx_LogLevel": "1"
```

```
                    },
                    "file_paths": {
                        "binaries": [
                            "/usr/bin/%%",
                            "/usr/sbin/%%"
                        ],
                        "configuration": [
                            "/etc/passwd",
                            "/etc/shadow"
                        ]
                    }
                },
                "is_default": false,
                "id": 8,
                "conditions": {
                    "os_name": {
                        "value": "*Ubuntu*"
                    },
                    "hostname": {
                        "value": "*ip*"
                    }
                },
                "description": "auto"
            }
}
```


Modifies a config for ID given

Modifies a config's name, queries, filters and auto config assign conditions.

URL: *https://<BASE_URL>/configs/<int:config_id>*

Request type: PUT

Example payload format:

```
{
        “name": “auto-config-linux",
        "queries": {
                "arp_cache": {
                        "id": 300,
                        "query": "select * from arp_cache;",
                        "interval": 86400,
                        "platform": "linux",
                        "snapshot": false,
                        "status": true
                }
        },
        "filters": {
                "events": {
                        "disable_subscribers": [
                                "user_events"
                        ]
                },
                "options": {
                        "custom_plgx_LogLevel": "1"
                },
                "file_paths": {
                        "binaries": [
                                "/usr/bin/%%",
                                "/usr/sbin/%%"
                        ],
                        "configuration": [
                                "/etc/passwd",
                                "/etc/shadow"
                        ]
                }
```

```
            },
            "is_default": false,
            "id": 8,
            "conditions": {
                    "os_name": {
                            "value": "*Ubuntu*"
                    },
                    "hostname": {
                            "value": "*ip*"
                    }
            },
            "description": "auto"
 }
```


Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
         "message": "Config is updated successfully for the platform given"
}
```


Deletes a config.

Deletes a config with ID given.

URL: *https://<BASE_URL>/configs/<int:config_id>*

Request type: DELETE

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status":"success",
        "message":"Config is deleted successfully"
}
```

Assigns a config.

Assigns a config to single/multiple node(s).

URL: *https://<BASE_URL>/configs /<int:config_id>/assign*

Request type: PUT

Example payload format:

```
{
        "host_identifiers": "EC2EBD16-8D48-0C24-EB42-B22333D7F08D, EC2EBD16-8D48-0C24-EB42-DBKYGSS ",
        "tags":""
}
```

Required payload arguments: host_identifiers / tags

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Config is assigned to the hosts successfully"
}
```

List of hosts assigned with the config.

Hosts list for which the config with given config id is assigned.

URL: *https://<BASE_URL>/configs/<int:config_id>/hosts*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
         "message": "Hosts are retried for the config",
         "data": [
                {
                        "id": 1,
                        "display_name": "ip-172-31-28-53",
                        "host_identifier": "ec21dee2-7085-d568-52f4-3c5b3f0710c7",
```

```
                "os_info": {
                    "name": "Ubuntu",
                    "build": "",
                    "major": "18",
                    "minor": "4",
                    "patch": "0",
                    "version": "18.04.5 LTS (Bionic Beaver)",
                    "codename": "bionic",
                    "platform": "ubuntu",
                    "platform_like": "debian"
                },
                "tags": [],
                "last_ip": "13.234.186.234",
                "is_active": false
            }
        ]
}
```


BLUEPRINT: alerts

Blueprint-path: /alerts

View alerts source distribution

Returns all alerts count for all the sources.

URL: *https://<BASE_URL>/alerts/count_by_source*

Request type: GET

Example payload format:


```
{
    "resolved": false,
    “duration”:”3”,
    “date”: "2020-8-5",
    “type”:”2”,
```

```
        "host_identifier": "EC2EBD16-8D48-0C24-EB42-B22333D7F08D",
        “rule_id”:1
}
```


Filters description:

duration – to get recent alerts by(month(4)/week(3)/day(2)/hr(1))

date – end date for the duration to be calculated by(format : 2020-10-14)

type – start date(1)/end date(2)

resolved – true to get only resolved alerts count / false to get non-resolved alerts count

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Data is fetched successfully",
        "data": {
                "alert_source": [
                {
                        "name": "virustotal",
                        "count": 0
                },
                {
                        "name": "rule",
                        "count": 93833
                },
                {
                        "name": "ibmxforce",
                        "count": 3
                },
                {
                        "name": "alienvault",
                        "count": 0
                },
                {
```

```
                    "name": "ioc",
                    "count": 21
                }
            ]
        }
}
```


View all alerts

Returns all alerts for the data filters applied.

URL: *https://<BASE_URL>/alerts*

Request type: POST

Example payload format:


```
{
    "source":"rule",
    "resolved":false,
    "start":0,
    “limit”:1,
    “searchterm”:”EC2”,
    “event_ids”: [],
    “duration”:”3”,
    “date”: "2020-8-5",
    “type”:”2”,
    “host_identifier”:””,
    “query_name”: "process_events",
    “rule_id”:2
}
```


Filters description:

source – alert’s source to get the alerts only for

resolved – true to get only resolved alerts / false to get non-resolved alerts

event_ids – event ids to filter the alerts for

duration – to get recent alerts by(month(4)/week(3)/day(2)/hr(1))

date – end date for the duration to be calculated by(format : 2020-10-14)

type – start date(1)/end date(2)

host_identifier – host identifier of the host to filter alerts by

query_name – query name to filter the alerts by

rule_id – id of the rule to filter the alerts by

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
    "status": "success",
    "message": "Data is fetched successfully",
    "data": {
      "count": 93833,
      "total_count": 93833,
      "results": [
            {
              "id": 93855,
              "node_id": 4,
              "rule_id": 146,
              "severity": "INFO",
              "rule": {
                    "name": "test_process_without_default_query",
                    "id": 146
              },
              "created_at": "2020-08-04 17:01:13.458996",
              "type": "rule",
              "source": "rule",
              "status": "OPEN",
              "alerted_entry": {
                    "eid": "241E415E-9F35-42DA-9F20-0D3F03F8FFFF",
                    "pid": "5704",
                    "path": "C:\\Windows\\System32\\wbem\\WmiPrvSE.exe",
                    "time": "1596557041",
                    "action": "PROC_TERMINATE",
```

```
            "cmdline": "C:\\Windows\\system32\\wbem\\wmiprvse.exe",
            "utc_time": "Tue Aug  4 16:04:01 2020 UTC",
            "owner_uid": "NT AUTHORITY\\NETWORK SERVICE",
            "parent_pid": "700",
            "parent_path": "C:\\Windows\\System32\\svchost.exe",
            "process_guid": "58E0F736-D62C-11EA-8283-02F7A50E7DFE",
            "parent_process_guid": "58E0F62F-D62C-11EA-8283-02F7A50E7DFE"
  },
  "hostname": "EC2AMAZ-H7M54UV",
  "aggregated_events_count": 20
},
{
  "id": 93854,
  "node_id": 4,
  "rule_id": 146,
  "severity": "INFO",
  "rule": {
            "name": "test_process_without_default_query",
            "id": 146
  },
  "created_at": "2020-08-04 17:01:13.448373",
  "type": "rule",
  "source": "rule",
  "status": "OPEN",
  "alerted_entry": {
            "eid": "74A4627E-AE27-4A1F-9DAC-2E6303F8FFFF",
            "pid": "5348",
            "path": "C:\\Windows\\WinSxS\\amd64_microsoft-windows-
      servicingstack_31bf3856ad364e35_10.0.17763.850_none_7e18264b4d00f4
      98\\TiWorker.exe",
            "time": "1596556952",
            "action": "PROC_CREATE",
```

```
                "cmdline": "C:\\Windows\\winsxs\\amd64_microsoft-windows-servicingstack_31bf3856ad364e35_10.0.17763.850_none_7e18264b4d00f498\\TiWorker.exe -Embedding",
                "utc_time": "Tue Aug  4 16:02:32 2020 UTC",
                "owner_uid": "NT AUTHORITY\\SYSTEM",
                "parent_pid": "700",
                "parent_path": "C:\\Windows\\System32\\svchost.exe",
                "process_guid": "58E0F73A-D62C-11EA-8283-02F7A50E7DFE",
                "parent_process_guid": "58E0F62F-D62C-11EA-8283-02F7A50E7DFE"
            },
            "hostname": "EC2AMAZ-H7M54UV",
            "aggregated_events_count": 20
        }
    ]
  }
}
```


Note:

Alerts generated from ESP can have one of the below 2 states. For now there is no automatic resolving alert based on investigation state, it should be done by the Admin manually once he resolved the alert on host

1. Active/Un-resolved state - Alerts in this state are ideally have not been investigated and fixed on agent.
2. Resolved state – Alerts in this state are investigated for the issues and have been fixed on agent.

Resolve/Unresolve alerts

Resolve/Unresolve alerts based on the payload parameter ‘resolve’ set. It just changes the status inbetween the above mentioned states.

URL: *https://<BASE_URL>/alerts*

Request type: PUT

Example payload format:


```
{
    “resolve": true,
    “alert_ids”: [1,2,3]
```

```
}
```


Required payload arguments: alert_ids

Filters description:

resolve – true to resolve / false to unresolve

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
    "status": "success",
    "message": "Alerts status is changed successfully"
}
```


View an alert

Returns an alert data.

URL: *https://<BASE_URL>/alerts/<int:alert_id>*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
    "status": "success",
    "message": "Successfully fetched the Alerts data",
    "data": {
        "query_name": "win_dns_events",
        "message": {
            "eid": "22682106-0532-4AA9-AEA6-6E6931000000",
            "pid": "1144",
            "time": "1596551122",
            "action": "DNS_LOOKUP",
            "utc_time": "Tue Aug  4 14:25:22 2020 UTC",
            "event_type": "DNS",
            "domain_name": ".www.google.com",
            "remote_port": "53",
            "request_type": "1",
```

```
                    "request_class": "1",
                    "remote_address": "172.31.0.2"
                },
                "node_id": 4,
                "rule_id": null,
                "severity": "WARNING",
                "created_at": "2020-08-04 15:36:49.651175",
                "type": "Threat Intel",
                "source": "ioc",
                "recon_queries": {},
                "status": "OPEN",
                "source_data": {},
                "hostname": "EC2AMAZ-H7M54UV",
                "platform": "windows"
        }
}
```


Export alerts

Exports alerts data.

URL: *https://<BASE_URL>/alerts/alert_source/export*

Request type: POST

Example payload format:


```
{
        “source": "rule”
}
```


Required payload arguments: source

Response: Returns a csv file.

View graph data

Returns alerts graph data.

URL: *https://<BASE_URL>/alerts/graph*

Request type: GET

Example payload format:

```
{
        "source":"rule",
        “duration”:"3",
        “date”:”2020-8-5”,
        “type”:”2”,
        "host_identifier": "EC2EBD16-8D48-0C24-EB42-B22333D7F08D",
        “rule_id”:1

}
```

Required payload arguments: source

Filters description:

source – alert’s source to get the alerts only for

duration – to get recent alerts by(month/week/day)

date – end date for the duration to be calculated by

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Data is fetched successfully",
        "data": [
                {
                        "start": 1596560473458.996,
                        "content": "",
                        "event_id": 93855,
                        "className": ""
                },
                {
                        "start": 1596560473448.373,
                        "content": "",
                        "event_id": 93854,
                        "className": ""
```

```
                }
        ]
}
```


View Related events of the host

Returns host's events related to the alert.

URL: *https://<BASE_URL>/alerts/<int:alert_id>/events*

Request type: GET

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
  "status": "success",
  "message": "Successfully fetched the Alert's events data",
  "data": {
          "schedule_query_data_list_obj": [
                  {
                    "name": "win_dns_events",
                    "data": [
                            {
                                    "eid": "15A348E7-AAD4-4099-8A53-F25532000000",
                                    "pid": "1144",
                                    "time": "1596551139",
                                    "action": "DNS_LOOKUP",
                                    "utc_time": "Tue Aug  4 14:25:39 2020 UTC",
                                    "event_type": "DNS",
                                    "domain_name": ".www.gstatic.com",
                                    "remote_port": "53",
                                    "request_type": "1",
                                    "request_class": "1",
                                    "remote_address": "172.31.0.2",
                                    "date": "Tue Aug  4 14:25:39 2020 UTC"
                            },
```

```
                        {
                                "eid": "579D64E5-8A48-4F87-83BB-F87B32000000",
                                "pid": "1144",
                                "time": "1596551139",
                                "action": "DNS_LOOKUP",
                                "utc_time": "Tue Aug  4 14:25:39 2020 UTC",
                                "event_type": "DNS",
                                "domain_name": ".www.gstatic.com",
                                "remote_port": "53",
                                "request_type": "1",
                                "request_class": "1",
                                "remote_address": "172.31.0.2",
                                "date": "Tue Aug  4 14:25:39 2020 UTC"
                        }
                 ]
                }
        ],
        "system_state_data_list": [
                "etc_hosts",
                "drivers",
                "kernel_info",
                "users",
                "scheduled_tasks",
                "uptime",
                "osquery_info",
                "os_version",
                "patches",
                "certificates"
        ]
}
}
```

View Aggregated events of the alert

Returns all events which are aggregated and related to the alert.

URL: *https://<BASE_URL>/alerts/<int:alert_id>/alerted_events*

Request type: POST

Example payload format:

```
{
        "query_name": "windows_events",
        "start":0,
        "limit":1,
        "searchterm":"windows",
        “column_name”: “md5”,
        “column_value”: “<Actual column value here>”
}
```

Response: Returns a JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully fetched the Alert's events data",
        "data": {
                "total_count": 1,
                "count": 1,
                “categorized_count”: 1,
                "results": [
                        {
                                "id": 5,
                                "columns": {
                                        "eid": "3C7713B7-F181-40CF-B875-A23A8FA8FFFF",
                                        "pid": "13304",
                                        "path": "C:\\Windows\\System32\\sc.exe",
                                        "time": "1618467246",
                                        "action": "PROC_CREATE",
                                        "cmdline": "sc  stop plgx_agent",
```

```
                "eventid": "2",
                "utc_time": "Thu Apr 15 06:14:06 2021 UTC",
                "owner_uid": "SHESHADRI\\Sheshadri D",
                "parent_pid": "8832",
                "parent_path": "C:\\Windows\\System32\\cmd.exe",
                "process_guid": "829157B0-9DA2-11EB-960F-E86A64EE0429",
                "parent_process_guid": "8291567E-9DA2-11EB-960F-E86A64EE0429"
            },
            "action": "added",
            "timestamp": "2021-04-15 09:03:29"
        }
    ]
}
}
```


Get alerts data for process analysis

Returns alerts data for process analysis.

URL: *https://<BASE_URL>/alerts/process*

Request type: POST

Example payload format:


```
{
    "process_guid": "58E0F736-D62C-11EA-8283-02F7A50E7DFE",
    "alert_id": "93855",
    "node_id": 4
}
```


Required payload arguments: process_guid and node_id

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
    "status": "success",
    "message": "Data is fetched successfully",
```

```
"data": {
        "name": "svchost.exe",
        "path": "C:\\Windows\\System32\\svchost.exe",
        "all_children": [
         {
          "action": "PROC_CREATE",
          "count": 4,
          "color": "blue",
          "node_type": "action",
          "children": [
           {
            "color": "red",
            "name": "child",
            "data": {
                    "eid": "44A8FC8A-2223-455B-89ED-BE1503F8FFFF",
                    "pid": "2156",
                    "path": "C:\\Windows\\System32\\taskhostw.exe",
                    "time": "1596551290",
                    "action": "PROC_CREATE",
                    "cmdline": "taskhostw.exe NGCKeyPregen",
                    "utc_time": "Tue Aug  4 14:28:10 2020 UTC",
                    "owner_uid": "NT AUTHORITY\\SYSTEM",
                    "parent_pid": "1032",
                    "parent_path": "C:\\Windows\\System32\\svchost.exe",
                    "process_guid": "58E0F688-D62C-11EA-8283-02F7A50E7DFE",
                    "parent_process_guid": "58E0F638-D62C-11EA-8283-
      02F7A50E7DFE"
           }
          }
         ],
          "last_time": "1596558693",
          "all_children": [
           {
```

```
                "color": "red",
                "name": "child",
                "data": {
                        "eid": "44A8FC8A-2223-455B-89ED-BE1503F8FFFF",
                        "pid": "2156",
                        "path": "C:\\Windows\\System32\\taskhostw.exe",
                        "time": "1596551290",
                        "action": "PROC_CREATE",
                        "cmdline": "taskhostw.exe NGCKeyPregen",
                        "utc_time": "Tue Aug  4 14:28:10 2020 UTC",
                        "owner_uid": "NT AUTHORITY\\SYSTEM",
                        "parent_pid": "1032",
                        "parent_path": "C:\\Windows\\System32\\svchost.exe",
                        "process_guid": "58E0F688-D62C-11EA-8283-02F7A50E7DFE",
                        "parent_process_guid": "58E0F638-D62C-11EA-8283-
                  02F7A50E7DFE"
                }
               }
              ],
               "name": "PROC_CREATE",
               "fetched": true,
               "process_guid": "58E0F638-D62C-11EA-8283-02F7A50E7DFE"
              }
             ],
             "node_type": "root",
             "data": {
                     "process_guid": "58E0F638-D62C-11EA-8283-02F7A50E7DFE",
                     "path": "C:\\Windows\\System32\\svchost.exe"
             }
        }
}
```


Get alerts data for process child analysis

Returns alerts data for process child analysis.

URL: *https://<BASE_URL>/alerts/process/child*

Request type: POST

Example payload format:


```
{
        "process_guid": "58E0F638-D62C-11EA-8283-02F7A50E7DFE",
        "action": "PROC_TERMINATE",
        "node_id": 4
}
```


Required payload arguments: process_guid, action and node_id

Response: Returns a JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully get the data",
        "data": {
                "child_data": [
                 {
                        "color": "red",
                        "name": "child",
                        "data": {
                                "eid": "404D77BA-055E-4EFD-9BE3-9C0403F8FFFF",
                                "pid": "2156",
                                "path": "C:\\Windows\\System32\\taskhostw.exe",
                                "time": "1596551290",
                                "action": "PROC_TERMINATE",
                                "cmdline": "taskhostw.exe NGCKeyPregen",
                                "utc_time": "Tue Aug  4 14:28:10 2020 UTC",
                                "owner_uid": "NT AUTHORITY\\SYSTEM",
                                "parent_pid": "1032",
                                "parent_path": "C:\\Windows\\System32\\svchost.exe",
                                "process_guid": "58E0F688-D62C-11EA-8283-02F7A50E7DFE",
```

```
                                "parent_process_guid": "58E0F638-D62C-11EA-8283-
                            02F7A50E7DFE"
                        }
                    }
                ],
                "last_time": "1596558693"
            }
}
```


Get the alerted host state

Returns the alerted host's state.

URL: *https://<BASE_URL>/alerts/<int:alert_id>/state*

Request type: GET

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the Alerted host state",
        "data": [
                {
                        "query_name": "uptime",
                        "count": 4
                },
                {
                        "query_name": "osquery_info",
                        "count": 1
                }
        ]
}
```


Exports similar alerted events for given alert id.

Returns CSV file for similar alerted events for given alert id.

URL: *https://<BASE_URL>/alerts/<int:alert_id>/alerted_events/export*

Request type: GET

Example payload format:

```
{
        "query_name":"win_file_events",
        "searchterm":"win",
        "column_name":"md5",
        "column_data":"<Actual column value here to filter>"
}
```

Required payload arguments: query_name

Response: Returns a csv file.

BLUEPRINT: response

Blueprint-path: /response

View all response actions

Returns all response actions.

URL: *https://<BASE_URL>/response*

Request type: POST

Example payload format:

```
{
        "start": 0,
        “limit”:1,
        “searchterm”:”pol”
}
```

Response: Returns JSON array of data, status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully fetched the responses info",
        "data": {
                "count": 1,
                "total_count": 1,
                "results": [
```

```
                {
                  "id": 7,
                  "action": "stop",
                  "command": {
                    "action": "stop",
                    "actuator": {
                      "endpoint": "polylogyx_vasp"
                    },
                    "target": {
                      "process": {
                        "name": "calc.exe",
                        "pid": "8282"
                      }
                    }
                  },
                  "created_at": "2020-08-04 15:10:11.284031",
                  "updated_at": "2020-08-04 15:10:11.283453",
                  "script_name": null,
                  "target": "process",
                  "Executed": "1/1"
                }
            ]
        }
}
```


View response actions in node level

Returns all response actions in node level.

URL: *https://<BASE_URL>/response/view*

Request type: POST

Example payload format:


```
{
    "start": 0,
```

```
        “limit”:1,
        “searchterm”:”EC2”,
        “openc2_id”:1
}
```


Required payload arguments: openc2_id

Response: Returns JSON array of data, status and message.

Example response format:


```
{
        "status": "success",
        "message": "Successfully fetched the responses info",
        "data": {
                "count": 1,
                "total_count": 7,
                "results": [
                 {
                        "id": 8,
                        "command": {
                         "action": "stop",
                         "actuator": {
                          "endpoint": "polylogyx_vasp"
                         },
                         "target": {
                          "process": {
                           "name": "calc.exe",
                           "pid": "8282"
                          }
                         }
                        },
                        "created_at": "2020-08-04 15:10:11.315204",
                        "updated_at": "2020-08-04 15:10:11.288915",
                        "node_id": 3,
                        "command_id": "2c92808273b870760173ba05d560000c",
```

```
                    "status": "failure",
                    "message": "RESP_SERVER_DISABLED",
                    "hostname": "EC2AMAZ-VLRC0S2",
                    "target": "process",
                    "action": "stop"
                }
            ]
        }
}
```


Export response actions

Returns all response actions into a csv file.

URL: *https://<BASE_URL>/response/export*

Request type: POST

Response: Returns a csv file.

View a response action

Returns all responses info for the command id given.

URL: *https://<BASE_URL>/response/<string:command_id>*

Request type: GET

Response: Returns JSON array of data, status and message.

Example response format:


```
{
    “status”:”success”,
    “message”:”Successfully received the command status”,
    “data”: {
            "id": 8,
            "command": {
              "action": "stop",
              "actuator": {
                "endpoint": "polylogyx_vasp"
              },
```

```
                "target": {
                  "process": {
                    "name": "calc.exe",
                    "pid": "8282"
                  }
                }
              },
              "created_at": "2020-08-04 15:10:11.315204",
              "updated_at": "2020-08-04 15:10:11.288915",
              "node_id": 3,
              "command_id": "2c92808273b870760173ba05d560000c",
              "status": "failure",
              "message": "RESP_SERVER_DISABLED",
              "hostname": "EC2AMAZ-VLRC0S2",
              "target": "process",
              "action": "stop"
        }
}
```


Get action status

Returns response action status of a host.

URL: *https://<BASE_URL>/response/status*

Request type: POST

Example payload format:


```
{
    “host_identifier”:”EC2CD1A0-140B-9331-7A60-CFFCE29D2E71”,
    “node_id”:1
}
```


Required payload arguments: host_identifier / node_id

Response: Returns JSON array of status, message and responseEnabled.

Example response format:


```
{
```

```
    "status": "success",
    "message": "Successfully received the status",
    "responseEnabled": true,
    "endpointOnline": true
}
```


Agent restart

Restart an agent.

URL: *https://<BASE_URL>/response/restart-agent*

Request type: POST

Example payload format:


```
{
    "host_identifier":"EC2CD1A0-140B-9331-7A60-CFFCE29D2E71"
}
```


Required payload arguments: host_identifier

Response: Returns JSON array of status and message.

Example response format:


```
{
    "status": "success",
    "message": "Action to restart agent is added successfully"
}
```


Add an action

Adds response action for the data given.

URL: *https://<BASE_URL>/response/add*

Request type: POST

Example payload format:

1)File Response Action:


```
{
    "action": "delete",
    "actuator_id": "EC2CD1A0-140B-9331-7A60-CFFCE29D2E71",
    "target": "file",
```

```
        "file_name": "C:\\Users\\PolyLogyx\\Downloads\\suspicious.exe",
        "file_hash": "o2MJjT8UKSRM7eoLDMWvm4LxqaFvDxd2wLg1KQQQ2jXfG5UE"
}
```


Required payload arguments: action, actuator_id, file_name/file_hash and target

2)Process Response Action:


```
{
        "action": "stop",
        "actuator_id": "EC2CD1A0-140B-9331-7A60-CFFCE29D2E71",
        "target": "process",
        "process_name": "suspicious1.exe",
        "pid": "3123"
}
```


Required payload arguments: action, actuator_id, process_name/pid and target

3)Network Response Action:

A) Delete a rule:


```
{
        "action": "delete",
        "actuator_id": "EC2CD1A0-140B-9331-7A60-CFFCE29D2E71",
        "target": "ip_connection",
        "rule_name": "test_rule_12"
}
```


Required payload arguments: action, actuator_id, rule_name and target

B) Isolate a rule:


```
{
        "action": "contain",
        "actuator_id": "EC2CD1A0-140B-9331-7A60-CFFCE29D2E71",
        "target": "ip_connection",
        "rule_name": "test_rule_12",
        "rule_group": "test",
        "src_port": "",
        "dst_port": "",
```

```
    "dst_addr": "",
    "application": "",
    "direction": "1",
    "layer4_protocol": "256"
}
```

Response: Returns JSON array of command_id, status and message.

Example response format:

```
{
    "status": "success",
    "message": “Successfully sent the response command”,
    “command_id”:"2c92808a69099f17016910516100000a"
}
```

Add a custom action:

Adds custom response action.

URL: *https://<BASE_URL>/response/custom-action*

Request type: POST

Example payload format:

```
{
    “host_identifier”:”EC2CD1A0-140B-9331-7A60-CFFCE29D2E71”,
    “content”:"dir\n$pwd",
    “file_type”:”4”,
    “save_script”:”true”,
    “script_name”:”dir_script”
}
```

Filters description:

file_type – 1 for .bat, 2 for powershell scripts and 3 for shell scripts

save_script - “true” to save the script, “false” not to save

Required payload arguments: host_identifier, file_type

Response: Returns JSON array of openc2_id, status and message.

Example response format:

```
{
```

```
        "status": "success",
        "message": “Action is added successfully”,
        “openc2_id”: 1
}
```

Delete a response action

Delete a response action record from the database.

URL: *https://<BASE_URL>/response/<int: openc2_id>/delete*

Request type: DELETE

Response: Returns JSON array of status and message.

Example response format:

```
{
        "status": "success",
        "message": "Successfully removed the response"
}
```

Cancelling Response:

Cancel response action.

URL: *https://<BASE_URL>/response/cancel*

Request type: POST

Example payload format:

```
{
        "command_id":"2c92808478ca277b0178ca2b9ef80005"
}
```

Required payload arguments: command_id

Response: Returns JSON array of openc2_id, status and message.

Example response format:

```
{
        "status" : 'success'
        "message": "Successfully sent command to cancel action"
}
```

Add live response:

Add live response action.

URL: *https://<BASE_URL>/response/live_response*

Request type: POST

Example payload format:

```
{
        "host_identifier":"EC2EBD16-8D48-0C24-EB42-B22333D7F08D",
        "content":"dir\n$pwd",
        "file_type":"2",
        "file":"",
        "save_script":"true",
        "script_name":"dir_script"
}
```

Required payload arguments: host_identifier, file_type

Response: Returns JSON array of openc2_id, status and message.

Example response format:

```
{
        "openc2_id": 133,
         "command_id": "2c92808478edeff20178ee8f18910025",
         "message": "Action is added successfully",
         "status": "success"
}
```

--> Make a connection from a socketio client to the below URL.

wss://<IP_OF_THE_SERVER>/websocket/action/result

--> Emit below payload to the socket server.

{“command _id”:<command_id _from_api_response>}

For ex: *{“command_id": "2c92808478edeff20178ee8f18910025"}*

--> Keep emitting the message ‘pong’ when ‘ping’ is received.

--> Keep the socket client listen to server till a message is received.

Get Response status of all hosts:

Returns host degraded status of all hosts.

URL: *https://<BASE_URL>/response/status/all*

Request type: GET

Response: Returns JSON array of openc2_id, status and message.

Example response format:


```
{
"status": "success",
        "message": "Successfully fetched the response action status of all hosts",
        "data": [
                {
                 "hostIdentifier": "ec2bff15-e870-567a-0ed2-3642bcbaf257",
                 "epName": "ip-172-31-10-241",
                 "onlineStatus": "ONLINE",
                 "responseEnabled": true,
                 "endpointOnline": true,
                 "host_degraded": false
                },
                {
                "hostIdentifier": "ec27bfa3-bbb1-f25f-f695-21ed19e5b62b",
                 "epName": "ip-172-31-17-167",
                 "onlineStatus": "ONLINE",
                "responseEnabled": true,
                 "endpointOnline": false,
                 "host_degraded": true
                 }
]
}
```


BLUEPRINT: defender-management

Blueprint-path: /defender-management

Scan-now action for Windows Defender.

To perform instant scan action for Windows Defender.

URL: *https://<BASE_URL>/defender-management/scan-now*

Request type: POST

Example payload format:

```
{
        "host_identifier": "EC27FE09-DEBD-F637-E17C-63CDCBF640DB",
        "scan_type": 1,
        "file_path":""
}
```

Required payload arguments: host_identifier, scan_type

scan_type - 1 - QuickScan

Scan_type - 2 – FullScan

Scantype - 3 – CustomScan(file_path should be provided to scan the path)

Response: Returns JSON array of data, status and message.

Example response format:

```
{
         "openc2_id": 77,
        "message": "Action is added successfully",
        "status": "success"
}
```

Configure action for Windows Defender.

To configure Windows Defender preferences by specifying exclusion path, file types and processes .

URL: *https://<BASE_URL>/defender-management/configure*

Request type: POST

Example payload format:

```
{
        "host_identifier": "EC27FE09-DEBD-F637-E17C-63CDCBF640DB",
        "file_type":".txt",
        "path":"c://",
        "process":"notepad.exe",
        "action":"add/remove"
```

}

Required payload arguments: host_identifier, action

Action – add – to add path, file type or process in exclusion

Action – remove – to remove existing path/filetype/process from exclusion

Response: Returns JSON array of data, status and message.

Example response format:

```
{
    "message": "Action is added successfully",
    "status": "success"
}
```

Scheduled scan for Windows Defender.

Scheduled scan for Windows defender.

URL: *https://<BASE_URL>/defender-management/schedule-scan*

Request type: POST

Example payload format:

```
{
    "host_identifier": "EC27FE09-DEBD-F637-E17C-63CDCBF640DB",
    "scan_type": 1,
    "time": "hh:mm",
    "scan_day":"21-04-2021"
}
```

Required payload arguments: host_identifier, scan_type, time

scan_type - 1 - QuickScan

Scan_type - 2 – FullScan

If scan type is 2 (Full scan) ,Scan day should be given

Scan day - 0 – Everyday

Scan day -1 – Sunday

Scan day - 2 – Monday

Scan day - 3 – Tuesday

Scan day - 4 – Wednesday

Scan day - 5 – Thursday

Scan day - 6 – Friday

Scan day - 7 – Saturday

Scan day - 8 – Never

Response: Returns JSON array of data, status and message.

Example response format:

```
{
    "openc2_id": 77,
    "message": "Action is added successfully",
    "status": "success"
}
```

Check updates for Windows Defender.

Checks and updates Windows Defender signatures if available.

URL: *https://<BASE_URL>/defender-management/check-update*

Request type: POST

Example payload format:

```
{
    "host_identifier": "EC27FE09-DEBD-F637-E17C-63CDCBF640DB"
}
```

Required payload arguments: host_identifier

Response: Returns JSON array of data, status and message.

Example response format:

```
{
    "openc2_id": 77,
    "message": "Action is added successfully",
    "status": "success"
}
```

Current-settings action for Windows Defender.

Add action to get the Windows Defender preferences.

URL: *https://<BASE_URL>/defender-management/current-settings*

Request type: POST

Example payload format:

```
{
        "host_identifier": "EC27FE09-DEBD-F637-E17C-63CDCBF640DB"
}
```

Required payload arguments: host_identifier

Response: Returns JSON array of data, status and message.

Example response format:

```
{
         "openc2_id": 77,
        "message": "Action is added successfully",
        "status": "success"
}
```

Computer-status action for Windows Defender.

Add action to get computer-status of Windows Defender.

URL: *https://<BASE_URL>/defender-management/computer-status*

Request type: POST

Example payload format:

```
{
        "host_identifier": "EC27FE09-DEBD-F637-E17C-63CDCBF640DB",
}
```

Required payload arguments: host_identifier

Response: Returns JSON array of data, status and message.

Example response format:

```
{
         "openc2_id": 77,
        "message": "Action is added successfully",
        "status": "success"
}
```

Add get-quarantine action for Windows Defender.

Add action to get quarantine threats detected by Windows Defender.

URL: *https://<BASE_URL>/defender-management/get-quarantine*

Request type: POST

Example payload format:

```
{
        "host_identifier": "EC27FE09-DEBD-F637-E17C-63CDCBF640DB",
}
```

Required payload arguments: host_identifier

Response: Returns JSON array of data, status and message.

Example response format:

```
{
         "openc2_id": 77,
        "message": "Action is added successfully",
        "status": "success"
}
```

Refresh status of Windows Defender.

Refresh status of Windows Defender Antivirus

URL: *https://<BASE_URL>/defender-management/status_refresh*

Request type: POST

Example payload format:

```
{
        "host_identifier": "EC27FE09-DEBD-F637-E17C-63CDCBF640DB",
}
```

Required payload arguments: host_identifier

Response: Returns JSON array of data, status and message.

Example response format:

```
{
         "message": "Defender status query has been sent",
        "status": "success"
}
```

Send carve query to the host.

Creates distributed query to carve the quarantine file from list

URL: *https://<BASE_URL>/defender-management/quarantine_file/carve*

Request type: POST

Example payload format:

```
{
        "host_identifier": "EC27FE09-DEBD-F637-E17C-63CDCBF640DB",
}
```

Required payload arguments: host_identifier

Response: Returns JSON array of data, status and message.

Example response format:

```
{
        "message": "carve query has been sent to the host",
        "status": "success"
}
```