PolyLogyx Endpoint Security Platform (ESP)

REST API Documentation

Table of Contents

1. Overview
2. REST based API
3. Versioning
4. Base URL
5. Authentication
6. Transport Security
7. Client Request Context
8. Pagination
9. Errors
10. Request Debugging
11. Terminology

Use Cases:

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

1. Overview

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

PolyLogyx REST API allows developers to use a programming language of their choice to integrate with the headless PolyLogyx 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 PolyLogyx 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. PolyLogyx 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 v0.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> and 5000 port with actual details for your PolyLogyx server. The Base URL follows this template: https://<server_ip>:5000/services/api/v0/

5. Authentication

The PolyLogyx 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 need 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 and does not automatically expire. If you believe your API key is compromised, 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 PolyLogyx server provides an unique API key called x-access-token, which is encoded by JWT mechanism and is used as an unique key for all API calls further. If no x-access-token or wrong x-access-token provided, it returns un-authorised error or API key error. X-access-token will be provided at the url https://<Base URL>/login .

Payload format is as below

```
{
“username”:”someusername”,
“password”:”passwordoftheuser”
}
```

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

PolyLogyx 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: PolyLogyx 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. Pagination

Requests that return a list of resources may support paging. Pagination is based on a cursor and not on page number.

9. 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
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

10. Request Debugging

The request ID 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”

11. Terminology

Fleet	Set of endpoints running the PolyLogyx agent and managed by the PolyLogyx server
Node	A specific endpoint that is actively monitored
Config	PolyLogxy osquery based agent derives its behavior from its configuration. The config is a JSON describing the various options used to instrument the agent behavior 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 effect its behavior. 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 PolyLogyx 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.

API - Section:

BLUEPRINT: general apis(common)
Blueprint-Path: /

1. User Login:

User logging in will be done here.

URL: https://<Base URL>/login
Request Type: POST
Example Payload Format:
{
“username”:”admin”,
“password”:”admin”
}
Response: Returns response of x-access-token value
Example Response Format:
{
“x-access-token”:
“eyJhbGciOiJIUzUxMiIsImlhdCI6MTU2NzY3MjcyMCwiZXhwIjoxNTY3NjczMzIwfQ.eyJpZCI6MX0.7jklhAly5ZO6xr1t0Y2ahkZvEEMnrescGK9nszqF-hMAProwbjOHaiRO3tBS5I2gdmVSqKqBHynveAFbmor7TA“
}

2. Change User Password:

Changes user’s password.

URL: https://<Base URL>/changepw
Request Type: POST
Example Payload Format:
{
“old_password”:”admin”,
“new_password”:”admin123”,
“confirm_new_password”:”admin123”
}
Response: Returns response of status and message
Example Response Format:
{
"status": "success",
"message": “password is updated successfully“
}

3. User Logout:

Makes user to logout and authentication end.

URL: https://<Base URL>/logout
Request Type: POST
Response: Returns response of status and message
Example Response Format:
{
"status": "success",
"message": “user logged out successfully“
}

4. Get a file from downloads path:

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

URL: https://<Base URL>/downloads/<path:filename>
Request Type: GET
ex: https://<Base URL>/downloads/certificate.crt
Response: Returns response of a file object

5. Get POLYLOGYX CPT file for a specific platform:

Returns a response of a POLYLOGYX CPT file object for a platform given.

URL: https://<Base URL>/cpt/<string:platform>
Request Type: GET
Response: Returns response of a file object

6. Get POLYLOGYX certificate file:

Returns a response of a POLYLOGYX certificate file object.

URL: https://<Base URL>/certificate
Request Type: GET
Response: Returns response of a file object

7. Update API keys:

Updates the API keys which are used in POLYLOGYX platform.

URL: https://<Base URL>/apikeys
Request Type:POST
Example Payload Format:
{
"IBMxForceKey":"304020f8-99fd-4a17-9e72-80033278810a",
"IBMxForcePass":"6710f119-9966-4d94-a7ad-9f98e62373c8",
"vt_key":"69f922502ee0ea958fa0ead2979257bd084fa012c283ef9540176ce857ac6f2c"
}

Response: Returns response of a status, data and message
Example Response Format:

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

8. View API keys:

Returns the API keys which are used in POLYLOGYX platform.

URL: https://<Base URL>/apikeys
Request Type:GET
Response: Returns response of a status, data and message
Example Response Format:

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

9. Update the options:

Modifies the options based on the data given.

URL: https://<Base URL>/options/add
Request Type: POST
Example Payload Format:

{"option":{
"custom_plgx_EnableLogging": "true",
"custom_plgx_LogFileName": "C:\\ProgramData\\
plgx_win_extension\\plgx-agent.log",
"custom_plgx_LogLevel": "1",
"custom_plgx_LogModeQuiet": "0",
"custom_plgx_ServerPort": "443",
"custom_plgx_enable_respserver": "true",
"schedule_splay_percent": 10
}}
Response: Returns response of data, status and message
Example Response Format:
{
"status": "success",
"message": “options are updated successfully“,
"data": {
"option":{
"custom_plgx_EnableLogging": "true",
"custom_plgx_LogFileName": "C:\\ProgramData\\
plgx_win_extension\\plgx-agent.log",
"custom_plgx_LogLevel": "1",
"custom_plgx_LogModeQuiet": "0",
"custom_plgx_ServerPort": "443",
"custom_plgx_enable_respserver": "true",
"schedule_splay_percent": 10
}
}}

10. Add Alerts through Hunt file upload:

Adds alerts based on the hunt file uploaded.

URL: https://<Base URL>/hunt-upload
Request Type: POST
Example Payload Format-1:
{
“file”: “hunt file object to add the alerts”,
“type”:”md5”
}
Example Response Format-1:
{
“status”: “success”,
“message”:”successfully fetched the data through the hunt file
uploaded”,
“data”:{
“EC2300D6-B0D5-F9A6-1237-6553106EC525”:
{“query_name”:”win_file_events”
“count”:4}
},
“EC241E83-BDC2-CAFC-BF9F-28C22B37A7F0”:

{“query_name”:”win_process_events”
“count”:6}
}
}

Example Payload Format-2:
{
“file”: “hunt file object to add the alerts”,
“type”:”md5”,
“host_identifier”:”EC2300D6-B0D5-F9A6-1237-6553106EC525”,
“query_name”:”win_file_events”,
“start”:2,
“limit”:10
}
Example Response Format-2:
{
"status": "success",
"message": "successfully fetched the data through the hunt file uploaded",
"data": [
{"eid": "04030A02-0BB2-4AD3-BCBE-317A03B8FFFF",
"md5": "b3215c06647bc550406a9c8ccc378756",
"pid": "5904",
"uid": "BUILTIN\\Administrators",
"time": "1564493377",
"action": "FILE_WRITE",
"hashed": "1",
"sha256":"c0de104c1e68625629646025d15a6129a2b4b6496",
"pe_file": "NO",
"utc_time": "Tue Jul 30 13:29:37 2019 UTC",
"target_path": "C:\\Users\\Administrator\\Downloads\\test\\5MB.zip",
"process_guid": "3D62F1B7-B2BC-11E9-824A-9313D46ED9F3",
"process_name": "C:\\Windows\\explorer.exe"
},
{"eid": "0A808224-88C6-453C-A469-D45703B8FFFF",
"md5": "44d88612fea8a8f36de82e1278abb02f",
"pid": "6524",
"uid": "BUILTIN\\Administrators",
"time": "1564497535",
"action": "FILE_RENAME",
"hashed": "1",
"sha256":"275a71899f7db9d1663fc695ec2fe2a2c4538”
"pe_file": "NO",
"utc_time": "Tue Jul 30 14:38:55 2019 UTC",
"target_path": "C:\\Users\\Administrator\\Downloads\\eicar.com.txt",
"process_guid": "3D62F2B2-B2BC-11E9-824A-9313D46ED9F3",
"process_name": "C:\\Program Files (x86)\\Google\\Chrome\\Application\

\chrome.exe"
}]
}

11. Search for data in result_log database table:

Searches for data in database tables and returns the data.

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
        }
}
```

Example Response Format-1:

```
{
“status”: “success”,
“message”:”successfully fetched the data through the payload given”,
“data”:{
        “EC2300D6-B0D5-F9A6-1237-6553106EC525”:
                {
                “query_name”:”win_file_events”
                “count”:4
                },
        “EC241E83-BDC2-CAFC-BF9F-28C22B37A7F0”:
                {
                “query_name”:”win_process_events”
                “count”:6
```

```
                }
}
```


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
}
```


Example Response Format-2:


```
{
"status": "success",
"message": "successfully fetched the data through the payload given",
"data": [
        {
        "name": "ec2_instance_tags",
        "interval": "3600",
        "wall_time": "256",
        "executions": "10",
        "output_size": "0",
        "avg_user_time": "8",
        "average_memory": "147997",
        "avg_system_time": "1"
        },
        {
        "name": "ec2_instance_tags",
        "interval": "3600",
```

```
        "wall_time": "410",
        "executions": "16",
        "output_size": "0",
        "avg_user_time": "6",
        "average_memory": "147997",
        "avg_system_time": "1"          }             ]
}
```


12. Delete query result for some recent days:

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
}
Response: Returns response of status and message
Example Response Format:
{
"status": "success",
"message": “query result data is deleted successfully“,
}

13. Export schedule query results into csv file:

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”
}
Response: Returns response of a file object

14. Export nodes info into csv file:

Returns a response of a csv file object with nodes info.

URL: https://<Base URL>/nodes_csv
Request Type: GET
Response: Returns response of a csv file object

BLUEPRINT: nodes
Blueprint-Path: /nodes

15. Get Node Info:

List currently managed nodes and their properties.

URL: https://<Base URL>/nodes/
Request Type: GET
Response: JSON Array of node and their properties, for e.g.
Example Response Format:


```
{
"status": "success",
"message": "nodes data fetched successfully",
"data": [
        {
        "id": 4,
        "host_identifier": "D8FC0C20-7D9A-11E7-9483-54E1AD6C8228",
        "node_info": {
                "computer_name": "DESKTOP-QIRBS33",
                "hardware_model": "80XL",
                "hardware_serial": "PF0UFSFS",
                "hardware_vendor": "LENOVO",
                "physical_memory": "8458571776",
                "cpu_physical_cores": "2"
                },
        "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",
                "platform_like": "windows"
        }
        "network_info": {
                "mac_address": "0a:00:27:00:00:06"
                },
        "node_key": "c6a054a5-ccac-42f2-b631-d1ba2fc59d8a",
        "last_checkin": "2019-04-08T06:32:27.355782",
        "enrolled_on": "2019-02-18T07:32:32.003949",
        "tags": [
                {
                "id": 1,
                 "value": "atul"
                },
                {
                "id": 9,
                "value": "t"
                },
```

```
                {
                        "id": 8,
                        "value": "a"
                }
                ]
        }
}
```


16. Get Node Info By Its host_identifier:

Lists a specific node info managed by the PolyLogyx server and its properties.

URL: https://<Base URL>/nodes/<string:host_identifier>
Request Type: GET
Response: A node with its properties.
Example Response Format:


```
{
"status": "success",
"message": "Successfully fetched the node info",
"data": {
        "system_data": {
                "system_data": []
        },
        "id": 6,
        "host_identifier": "216F6B87-8922-4BAE-A68A-0E5EB11ACA1C",
        "node_info": {
                "computer_name": "Moulik",
                "hardware_model": "Virtual Machine",
                "hardware_serial": "0000-0002-4092-6255-5531-
9878-86",
                "hardware_vendor": "Microsoft Corporation",
                "physical_memory": "3757625344",
                "cpu_physical_cores": "1"
        },
        "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",
                "platform_like": "windows"
        }
        "network_info": {
                "mac_address": "00:00:00:00:00:00:00:e0"
        },
        "node_key": "d049a880-445a-4e20-8458-8752d28ec940",
```

```
        "last_checkin": "2019-04-03T16:37:32.522727",
        "enrolled_on": "2019-02-20T06:31:56.574275",
        "tags": [
                {
                "id": 3,
                "value": "moulik"
                }
                ]
        }
}
```


17. Edit Tags of a specific Node:

Edits tags of a Node for the host_identifier given.

URL: https://<Base URL>/nodes/tag/edit
Request Type: POST
Example Payload Format:


```
{
“host_identifier”:””216F6B87-8922-4BAE-A68A-0E5EB11ACA1C”,
“add_tags”: [“plgx, ”sample”],
“remove_tags”: [“simple”, “aaa”]
}
```


Response: Returns a response json containing status and message.
Example Response Format:


```
{
        “status”: ”success”,
        “message”: “Successfully modified the tag(s)”
}
```


18. Get schedule queries list for a Node:

Returns all schedule queries data of a specific node for the host_identifier given.

URL: https://<Base URL>/nodes/schedule_query/<string:host_identifier>
Request Type: GET
Response: Returns a response json containing data, status and message.
Example Response Format:


```
{
"status": "success",
"message": "Successfully received schedule query results",
"data": [{
        "id": 3010586,
        "name": "win_dns_response_events",
        "timestamp": "05/23/2019 07/10/02",
        "action": "added",
        "columns": {
```

```
                    "eid": "0BC3B847-FFF9-407F-87EA-430D16000000",
                    "pid": "1336",
                    "time": "1558595375",
                    "action": "",
                    "utc_time": "Thu May 23 07:09:35 2019 UTC",
                    "event_type": "DNS_RESPONSE",
                    "domain_name": ".ec2messages.ap-south-
1.amazonaws.com",
                    "remote_port": "53",
                    "resolved_ip": "52.95.88.152",
                    "request_type": "1",
                    "request_class": "1",
                    "remote_address": "172.31.0.2"
                    },
            "node_id": 12
            }}
```


19. Get Schedule Queries Result for a Node:

Returns schedule query results of a Node for the host_identifier given.

URL: https://<Base URL>/nodes/schedule_query/results
Request Type: POST
Example Payload Format:


```
{
“host_identifier”:””216F6B87-8922-4BAE-A68A-0E5EB11ACA1C”,
“query_name”: “win_file_events”,
“start”: 1,
“limit”: 20
}
```


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


```
{
        “status”: ”success”,
        “message”: “Successfully received node schedule query results”
        “data”: [{
                "id": 4439993,
                "name": "win_dns_response_events",
                "timestamp": "2019-06-07T14:52:11",
                "action": "added",
                "columns": {
                        "eid": "3B7C7A62-6D3C-404D-924C-
E77F51000000",
                        "pid": "1308",
                        "time": "1559919087",
                        "action": "",
                        "utc_time": "Fri Jun  7 14:51:27 2019 UTC",
                        "event_type": "DNS_RESPONSE",
                        "domain_name": ".ec2messages.ap-south-
```

```
1.amazonaws.com",
                        "remote_port": "53",
                        "resolved_ip": "52.95.80.172",
                        "request_type": "1",
                        "request_class": "1",
                        "remote_address": "172.31.0.2"
                },
                "node_id": 16,
                "node": {
                        "id": 16,
                        "host_identifier": "EC2A1F1D-0C6E-072D-C830-
392246FCBAAE",
                        "node_key": "9c7a7086-8f0f-4d45-abd2-
68b1d3149439",
                        "last_checkin": "2019-06-13T12:01:32.839308",
                        "enrolled_on": "2019-04-23T09:52:43.761165",
                        "tags": [
                                {
                                "id": 5,
                                "value": "Windows"
                                }
                                ]
                        }
                }]
        }
```


20. Get List Of Tags for a Node:

Returns list of tags of a Node for the host_identifier given.

URL: https://<Base URL>/nodes/<string:host_identifier>/tags
Request Type: GET
Response: Returns a response json containing data, status and message.
Example Response Format:


```
{
        “status”: ”success”,
        “message”: “Successfully fetched the tag(s)”
        “data”: [
                {
                "id": 1,
                "value": "atul"
                },
                {
                "id": 9,
                "value": "t"
                }
                ]
        }
```

21. Create Tags To a Node:

Creates tags to a node.

URL: https://<Base URL>/nodes/<string:host_identifier>/tags
Request Type: POST
Example Payload Format:

```
{
“tags”:[“Atul”, “Mould”]
}
```

Response: Returns a response json containing status and message.
Example Response Format:

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

22. Search Export To CSV File:

Exports the search result into csv file.

URL: https://<Base URL>/nodes/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”
```

}
Response: Returns a response of a File Object.

23. Get Query Results of a Node:

Returns query results of a node.

URL: https://<Base URL>/nodes/<string:host_identifier>/queryResult
Request Type: POST
Example Payload Format:
{
"start":1,
"length":100,
"search[value]":"5ad7fff3-cef4-4d"
}
Response: Returns a json response containing data, status and message
Example Response Format:
{
“status”: “success”,
“message”: “Query results are fetched successfully”,
“data”: ["certificates": {
"iRecordsFiltered": "0",
"iTotalRecords": "29",
"pageLength": "100",
"iTotalDisplayRecords": "0",
"aaData": []
},
"chrome_extensions": {
"iRecordsFiltered": "0",
"iTotalRecords": "18",
"pageLength": "100",
"iTotalDisplayRecords": "0",
"aaData": []
}]
}

24. Get Activity Results of a Node:

Returns activity results of a node.

URL: https://<Base URL>/nodes/<string:host_identifier>/activity
Request Type: POST
Example Payload Format:
{
“timestamp”:”Jun 1 2005 1:33PM”
}
Response: Returns a json response containing data, status and message

Example Response Format:


```
{
"status": "success",
"message": "Node activity is fetched successfully",
"data": {
        "node": {
        "host_identifier": "EC2A1F1D-0C6E-072D-C830-392246FCBAAE",
        "node_key": "9c7a7086-8f0f-4d45-abd2-68b1d3149439"
        },
        "queries_packs": [
        "certificates",
        "chrome_extensions",
        "drivers",
        "kernel_info",
        "Listening_ports",
        "osquery_info",
        "os_version",
        "pack/vuln-management/chrome_extensions",
        "patches",
        "scheduled_tasks",
        "uptime",
        "users",
        "win_dns_events",
        "win_dns_response_events",
        "win_http_events",
        "win_pefile_events",
        "win_process_events",
        "win_registry_events",
        "win_socket_events",
        "win_ssl_events"
        ]
        }
}
```


BLUEPRINT: tags
Blueprint-Path: /tags

25. Get List of all Tags:

Returns list of all tags.

URL: https://<Base URL>/tags/
Request Type: GET
Response: Returns a json response containing data, status and message
Example Response Format:


```
{
        “status”: “success”,
        “message”: “node activity is fetched successfully”,
```

```
            "data": [
                    {
                    "value": "atul",
                    "nodes": [
                            {
                            “node_key”:”c6a054a5-ccac-42f2-b631-
d1ba2fc59d8a”,
                            “host_identifier”:”D8FC0C20-7D9A-11E7-9483-
54E1AD6C8228"
                            }
                            ],
                    "packs": [],
                    "queries": [
                            {
                            "name": "Windows Defender Detections",
                            "sql": "select * from windows_events where data
like '%Detection%';"
                            },
                            {
                            "name": "FIM query",
                            "sql": "select * from win_file_events;"
                            }
                            ],
                    "file_paths": []
                    }
        }
```

26. Add List of all Tags to Tag table:

Adds list of all tags to Tag table.

URL: https://<Base URL>/tags/add
Request Type: POST
Example Payload format:

```
{
“tags”:[“Atul”,”moulik”]
}
```

Response: Returns a json response containing status and message
Example Response Format:

```
{
        “status”: “success”,
        “message”: “Tags are added successfully”,
}
```

BLUEPRINT: alerts
Blueprint-Path: /alerts

27. View Alerts Info:

Returns an alert’s info for the data given.

URL: https://<Base URL>/alerts/
Request Type: POST
Example Payload Format:

```
{
"host_identifier":"77858CB1-6C24-584F-A28A-E054093C8924",
"query_name":"processes",
"rule_id":3
}
```

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

```
{
"data": [ {
        "created_at": "Tue, 31 Jul 2018 14:19:30 GMT",
        "id": 1,
        "message": {
                "cmdline": "/sbin/launchd",
                "cwd": "/",
                "egid": "0",
                "euid": "0",
                "gid": "0",
                "name": "launchd",
                "nice": "0",
                "on_disk": "1",
                "parent": "0",
                "path": "/sbin/launchd",
                "pgroup": "1",
                "pid": "1",
                "resident_size": "6078464",
                 "root": "",
                "sgid": "0",
                "start_time": "0",
                "state": "R",
                "suid": "0",
                "system_time": "105116",
                "threads": "4",
                "total_size": "17092608",
                "uid": "0",
                "user_time": "10908”,
                "wired_size": “0”
                }
        "node_id": 1,
        "query_name": "processes",
        "rule_id": 3,
        "sql": null
        }]
"message": "Successfully received the alerts",
"status": "success"
}
```

28. Get Alerts Data:

Returns alert data.

URL: https://<Base URL>/alerts/data/<int:alert_id>
Request Type: GET
Response: Returns a json response containing data, status and message
Example Response Format:


```
{
"status": "success",
"message": "data is fetched successfully",
"data": {
        "distributed_query_tasks": [],
        "schedule_query_data": [
                                {
                                "name": "win_file_events",
                                "data": []
                                }],
        "alert": {
        "query_name": "Windows Defender Detections",
        "message": "{'eid': '2F9F7449-51A7-11E9-8043-484520FA5F27',
'pid': '24572',
                        'path': 'C:\\\\Windows\\\\System32\\\\sethc.exe',
                        'time': '1553811447',
                        'action': 'PROC_CREATE',
                        'cmdline': 'sethc.exe 211',
                        'utc_time': 'Wed Apr 03 22:17:27 2019 UTC',
                        'owner_uid': 'SJ-ASUS-LAPTOP2\\\\sjayanthi',
                        'parent_pid': '16988',
                        'parent_path': 'C:\\\\Windows\\\\System32\\\\
winlogon.exe',
                        'process_guid': '2F9F744A-51A7-11E9-8043-
484520FA5F27',
                        'parent_process_guid': 'B7815CB6-50E6-11E9-8043-
484520FA5F27'}",
                        "node_id": "1",
                        "rule_id": "29",
                        "severity": "WARNING",
                        "sql": null,
                        "created_at": "2019-04-03T11:31:44.836304",
                        "recon_queries": {
                                "scheduled_queries": [
                                        {
                                        "name": "win_file_events",
                                        "before_event_interval": 30,
                                        "after_event_interval": 60
                                        }
                                        ]
```

```
                    },
            "result_log_uid": "93c41a21-933e-4983-9c87-6020b4824719",
            "type": "rule",
            "source": "rule",
            "source_data": {}
            },
            "node": {
            "id": 1,
            "host_identifier": "9E1ADEF0-F00D-7840-8AF5-BC8E5E6B60E2",
            "node_key": "ee3965da-66f8-4b7f-929e-40212c7ffd4e",
                    ]
            }
            }
    }
```


BLUEPRINT: carves
Blueprint-Path: /carves

29. Get Carves:

Returns Carves.

URL: https://<Base URL>/carves/
Request Type: POST
Example Payload Format:


```
{
"host_identifier":"77858CB1-6C24-584F-A28A-E054093C8924"
}
```


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


```
{       "data": [ {
            "archive": "2N1P2UNDY6cd0877fa-36e4-41ff-926a-ff2a22673dc3.tar",
            "carve_guid": "cd0877fa-36e4-41ff-926a-ff2a22673dc3",
            "created_at": "2018-07-24 07:50:05",
            "id": 10,
            "block_count": 1,
            "node_id": 1,
            "session_id": "2N1P2UNDY6"
            "carve_size": 5632,
            }]
    "message": "Successfully fetched the carves”,
    "status": "success"
    }
```


30. Download Carves:

Returns a file object of Carves.

URL: https://<Base URL>/carves/download/<string:session_id>
Request Type: GET
Response: Returns a file object of node carves

BLUEPRINT: distributed
Blueprint-Path: /distributed

31. Add Distributed 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"],
        "description":"njkbgdf"
}
```

Response: Returns response of query_id, status and message
Example Response Format:

```
{
"status": "success",
"message": "Successfully send the distributed query",
"query_id":6
}
```

BLUEPRINT: yara
Blueprint-Path: /yara

32. List YARA files:

Returns list of yara file names.

URL: https://<Base URL>/yara/
Request Type: GET
Response: Returns response of data, status and message
Example Response Format:

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

BLUEPRINT: iocs

Blueprint-Path: /iocs

33. List IOC files:

Returns list of ioc file names.

URL: https://<Base URL>/iocs/
Request Type: GET
Response: Returns response of data, status and message
Example Response Format:

```
{
"status": "success",
"message": "Successfully fetched the ioc files",
"data": [
        {
        "type": "hello_name",
        "intel type": "self",
        "value": "dummy_testing_rr.com",
        "threat name": "test-intel_domain_name"
        }]
}
```

34. Add IOC files:

Upload ioc file.

URL: https://<Base URL>/iocs/add
Request Type: POST

```
{
“file”:”a file object here”
}
```

Response: Returns response of status and message
Example Response Format:

```
{
"status": "success",
"message": "Successfully updated the intel from the file uploaded“
}
```

BLUEPRINT: email
Blueprint-Path: /email

35. Configure Email:

configures email data like recipients, sender, smtp port.

URL: https://<Base URL>/email/configure
Request Type: POST
Example Payload Format:

{
"emailRecipients": ["mehtamouli1k@gmail.com", "moulik1@polylogyx.com"],
"email": "mehtamoulik13@gmail.com",
"smtpAddress": "smtp2.gmail.com",
"password": "a", "smtpPort": 445
}
Response: Returns response of data, status and message
Example Response Format:
{
"status": "success",
"message": "Successfully updated the email configuration“,
“data”:{
"emalRecipients": ["mehtamouli1k@gmail.com", "moulik1@polylogyx.com"]
"email": "mehtamoulik13@gmail.com",
"smtpAddress": "smtp2.gmail.com",
"emails": "mehtamoulik@gmail.com,moulik@polylogyx.com",
"password": "YQ==\n", "smtpPort": 445
}
}

BLUEPRINT: schema
Blueprint-Path: /schema

36. Get Schema:

Returns all tables schema.

URL: https://<Base URL>/schema
Request Type: GET
Response: Returns response of data, status and message
Example Response 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)",
}
}

37. Get Table Schema:

Returns a table schema for the table name given.

URL: https://<Base URL>/schema/<string:table>
Request Type: GET
Response: Returns response 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

38. Get All Rules info:

Returns all rules info.

URL: https://<Base URL>/rules/
Request Type: GET
Response: Returns response of data, status and message
Example Response Format:
{
"status": "success",
"message": "successfully fetched the rules info",
"data": [
{
"id": 2,
"alerters": [
"email",
"debug"
],
"conditions": "{'rules':
[{'id': 'column',
'type': 'string',
'field': 'column',
'input': 'text',
'value': ['target_name', '\\\\services\\\\Netlogon\\\\Parameters\\\\DisablePasswordChange'],
'operator': 'column_contains'
},
{'id': 'column',
'type': 'string',
'field': 'column',
'input': 'text',
'value': ['action', 'REG_SETVALUE'],

```
                                'operator': 'column_equal'
                        }],
                        'valid': True,
                        'condition': 'AND'}",
                "description": "table : win_registry_events. Hit when
target_name contains \\services\\Netlogon\\Parameters\\
DisablePasswordChange",
                "name": "MA_T0000_disable_password_change",
                "status": "ACTIVE",
                "updated_at": "2019-02-18T07:32:37.164541”,
                "type": "MITRE",
                "tactics": [
                        "persistence",
                        "defense-evasion"
                        ],
                "technique_id": "T1197"
                }
                ]
        }
```


39. Get A Rule info:

Returns a rule info for the id given.

URL: https://<Base URL>/rules/<int:rule_id>
Request Type: GET
Response: Returns response of data, status and message
Example Response Format:


```
{
"status": "success",
"message": "successfully fetched the rules info",
"data": {
        "id": 2,
        "alerters": [
                        "email",
                        "debug"
                        ],
                "conditions": "{'rules': [{'id': 'column', 'type': 'string', 'field':
'column', 'input': 'text', 'value': ['target_name', '\\\\services\\\\Netlogon\\\\
Parameters\\\\DisablePasswordChange'], 'operator': 'column_contains'}, {'id':
'column', 'type': 'string', 'field': 'column', 'input': 'text', 'value': ['action',
'REG_SETVALUE'], 'operator': 'column_equal'}], 'valid': True, 'condition':
'AND'}",
                "description": "table : win_registry_events. Hit when
target_name contains \\services\\Netlogon\\Parameters\\
DisablePasswordChange",
                "name": "MA_T0000_disable_password_change",
                "status": "ACTIVE",
                "updated_at": "2019-02-18T07:32:37.164541”,
                "type": "MITRE",
```

```
        "tactics": [
        "persistence",
        "defense-evasion"
                ],
        "technique_id": "T1197"
        }
}
```

40. Edit A Rule info:

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

URL: https://<Base URL>/rules/<int:rule_id>
Request Type: POST
Example Payload Format:

```
{
"alerters": [
"email",
"debug"
],
"conditions": {"rules":
                        [{"id": "column",
                        "type": "string",
                        "field": "column",
                        "input": "text",
                        "value": ["target_name", "\\\\services\\\\Netlogon\\\\
Parameters\\\\DisablePasswordChange"],
                        "operator": "column_contains"
                        },
                        {"id": "column",
                        "type": "string",
                        "field": "column",
                        "input": "text",
                        "value": ["action", "REG_SETVALUE"],
                        "operator": "column_equal"}],
                "valid": true,
                "condition": "AND"},
        "description": "table : win_registry_events. Hit when target_name
contains \\services\\Netlogon\\Parameters\\DisablePasswordChange",
        "name": "kishorebckajv6”,
        "type": "MITRE",
        "tactics": [
                "persistence",
                "defense-evasion"
        ],
        "technique_id": "T1197"
        }
```

Response: Returns response of data, status and message
Example Response Format:

```
{
```

```
"status": "success",
"message": “Successfully modified the rules info",
"data": {
        "id": 2,
        "alerters": [
                        "email",
                        "debug"
                        ],
        "conditions": {"rules":
                        [{"id": "column",
                        "type": "string",
                        "field": "column",
                        "input": "text",
                        "value": ["target_name", "\\\\services\\\\Netlogon\\\\
Parameters\\\\DisablePasswordChange"],
                        "operator": "column_contains"
                        },
                        {"id": "column",
                        "type": "string",
                        "field": "column",
                        "input": "text",
                        "value": ["action", "REG_SETVALUE"],
                        "operator": "column_equal"}],
                "valid": true,
                "condition": "AND"},
        "description": "table : win_registry_events. Hit when
target_name contains \\services\\Netlogon\\Parameters\\
DisablePasswordChange",
        "name": "MA_T0000_disable_password_change",
        "status": "ACTIVE",
        "updated_at": "2019-02-18T07:32:37.164541"
        }
}
```


41. Add A Rule:

Adds a rule info for the data given.

URL: https://<Base URL>/rules/add
Request Type: POST
Example Payload Format:


```
{
"alerters": [
"email",
"debug"
],
"conditions": {"rules":
                        [{"id": "column",
                        "type": "string",
                        "field": "column",
```

```
                                "input": "text",
                                "value": ["target_name", "\\\\services\\\\Netlogon\\\\
Parameters\\\\DisablePasswordChange"],
                                "operator": "column_contains"
                                },
                                {"id": "column",
                                "type": "string",
                                "field": "column",
                                "input": "text",
                                "value": ["action", "REG_SETVALUE"],
                                "operator": "column_equal"}],
                        "valid": true,
                        "condition": "AND"},
    "description": "table : win_registry_events. Hit when target_name
contains \\services\\Netlogon\\Parameters\\DisablePasswordChange",
        "name": “poly_rule”,
        "type": "MITRE",
        "tactics": [
                "persistence",
                "defense-evasion"
        ],
        "technique_id": "T1197"
        }
```

Response: Returns response of rule_id, status and message
Example Response Format:

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


BLUEPRINT: queries
Blueprint-Path: /queries

42. Get All Queries info:

Returns all queries info.

URL: https://<Base URL>/queries/
Request Type: GET
Response: Returns response of data, status and message
Example Response Format:

```
{
"status": "success",
"message": "successfully fecthed the queries info",
"data": {
        "id": 1,
        "name": "win_file_events",
        "sql": "select * from win_file_events;",
```

```
        "interval": 13,
        "platform": "windows",
        "version": "2.9.0",
        "description": "Windows File Events",
        "value": "File Events",
        "snapshot": true,
        "shard": null,
        "packs": [
         "<Pack: all-events-pack>"
        ]
        }
}
```


43. Get A Query info through query_id:

Returns a query info for the id given.

URL: https://<Base URL>/queries/<int:query_id>
Request Type: GET, POST
Response: Returns response of data, status and message
Example Response Format:


```
{
"status": "success",
"message": "successfully fecthed the query info for the given id",
"data": {
        "id": 1,
        "name": "win_file_events",
        "sql": "select * from win_file_events;",
        "interval": 13,
        "platform": "windows",
        "version": "2.9.0",
        "description": "Windows File Events",
        "value": "File Events",
        "snapshot": true,
        "shard": null,
        "packs": [
         "<Pack: all-events-pack>"
        ]
        }
}
```


44. 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"]
}
Response: Returns response of query_id, status and message
Example Response Format:
{
"status": "success",
"message": “Successfully added the query for the data given”,
“query_id”: 2
}

45. Edit Tags of a Query:

modifies tags for a query for id given.

URL: https://<Base URL>/queries/tag/edit
Request Type: POST
{
“query_id”: 1,
“add_tags”: ["finance12”,”sales12”],
“remove_tags":["finance","sales"]
}
Response: Returns response of status and message
Example Response Format:
{
"status": "success",
"message": “Successfully modified the tag(s)”
}

46. List Tags of a Query:

Returns tags of a query for id given.

URL: https://<Base URL>/queries/<int:query_id>/tags
Request Type: GET
Response: Returns response of data, status and message
Example Response Format:
{
"status": "success",
"message": “Successfully fetched the tag(s)”,
“data”:[“finance","sales"]
}

47. 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:
{
“tags":["finance","sales"]
}
Response: Returns response of status and message
Example Response Format:
{
"status": "success",
"message": “Successfully created the tag(s) to queries”
}

BLUEPRINT: packs
Blueprint-Path: /packs

48. Get All Packs info:

Returns all Packs info.

URL: https://<Base URL>/packs/
Request Type: GET
Response: Returns response of data, status and message
Example Response Format:


```
{
"status": "success",
"message": "successfully fetched the packs info",
"data": [
        {
        "id": 3,
        "name": "forensic-pack",
        "platform": null,
        "version": null,
        "description": null,
        "shard": null,
        "queries": [
                {
                "name": "auto exec",
                "sql": "select * from auto_exec;"
                },
                {
                "name": "appcompat_shims",
                "sql": "select * from appcompat_shims;"
```

```
                        }
                ]
        }
        ]
}
```

49. Get A Pack info:

    Returns a Pack info for the id given.

    URL: https://<Base URL>/packs/<int:pack_id>
    Request Type: GET
    Response: Returns response of data, status and message
    Example Response Format:

```
{
"status": "success",
"message": "successfully fetched the packs info",
"data": [
        {
        "id": 3,
        "name": "forensic-pack",
        "platform": null,
        "version": null,
        "description": null,
        "shard": null,
        "queries": [
                        {
                        "name": "auto exec",
                        "sql": "select * from auto_exec;"
                        },
                        {
                        "name": "appcompat_shims",
                        "sql": "select * from appcompat_shims;"
                        }
                ]
        }
        ]
}
```

50. 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"] }
```


Response: Returns response of pack_id, status and message
Example Response Format:


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


51. Edit tags of A Pack:

Edit tags of a Pack.

URL: https://<Base URL>/packs/tag/edit
Request Type: POST
Example Payload Format:


```
{
“pack_id”: 2,
“add_tags":["finance","sales”],
“remove_tags":["finance12”,”sales12”]
}
```


Response: Returns response of status and message
Example Response Format:


```
{
“status”:”success”,
“message”:”Successfully modified the tag(s)”,
}
```


52. List tags of A Pack:

Returns list of tags of a Pack for the name given.

URL: https://<Base URL>/packs/<string:pack_name>/tags
Request Type: GET
Response: Returns response of data, status and message
Example Response Format:


```
{
“status”:”success”,
“message”:”Successfully fetched the tag(s)”,
“data”: [“tag1”, “tag2”]
```

}

53. Add tags to A Pack:

Adds tags to a Pack for the name given.

URL: https://<Base URL>/packs/<string:pack_name>/tags
Request Type: POST
Example Payload Format:
{
“tags”: [“tag1”, “tag2”]
}
Response: Returns response of data, status and message
Example Response Format:
{
“status”:”success”,
“message”:”Successfully created the tag(s) to packs”,
}

54. Add packs through file upload:

Adds packs through a file upload.

URL: https://<Base URL>/packs/upload
Request Type: POST
Example Payload Format:
{
“file”: “ A JSON file object with json content same as /packs/add”
}
Response: Returns response of pack_id, status and message
Example Response Format:
{
“status”:”success”,
“message”:”pack uploaded successfully”,
“pack_id”:2
}