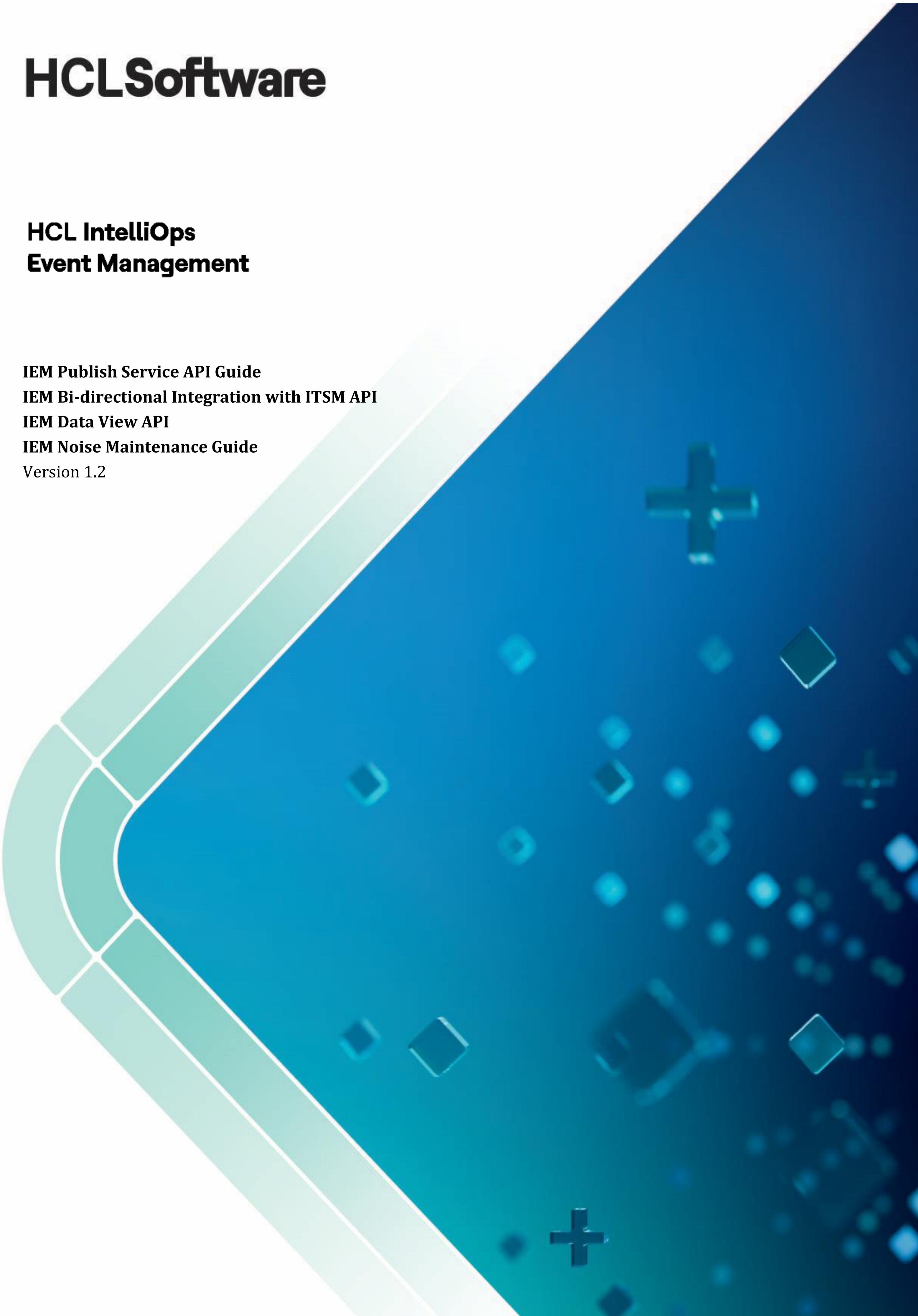
HCLSoftware

HCL IntelliOps
Event Management

IEM Publish Service API Guide
IEM Bi-directional Integration with ITSM API
IEM Data View API
IEM Noise Maintenance Guide
Version 1.2



The data contained in this document shall not be duplicated, used, or disclosed as a whole or in part for any purpose. If a contract is awarded to chosen parties because of or in connection with the submission of this data, the client or prospective client shall have the right to duplicate, use, or disclose this data to the extent provided in the contract. This restriction does not limit the client's or prospective client's right to use the information contained in the data if it is obtained from another source without restriction. The data subject to this restriction is contained in all marked sheets.

HCL has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the HCL website at www.hcltechsw.com.

Table of Contents

1 **Preface** 10

1.1 Intended Audience 10

1.2 Conventions 10

2 **IEM API Overview** 11

2.1 IEM Publish Service Overview 11

2.2 IEM Bi-directional Integration Overview 11

3 **About IEM Publish Service APIs** 12

3.1 IEM Publish Service Rest API Request 12

3.1.1 Authentication 12

3.1.2 Request Header 12

3.2 IEM Publish Service Rest API Responses 13

3.2.1 HTTP Response Codes 13

3.3 How to call IEM Publish Service APIs 13

3.3.1 Calling Token API to get a token 13

3.3.2 Calling Publish API endpoint with access token 14

3.4 IEM Publish Service API Details 19

3.4.1 Generate Token 20

3.4.2 API: Publish Events Data 21

3.4.3 API: Publish Metrics Data 22

3.4.4 API: Publish Entity Topology Data 24

3.4.5 API: Publish Service Topology Data 25

3.4.6 API: Publish Healthcheck Data 27

4 **About IEM Bi-directional Integration with ITSM APIs** 29

4.1 IEM Bi-directional Integration with ITSM Rest API Request 29

4.1.1 Authentication 29

4.1.2 Request Header 29

4.2 IEM Bi-directional Integration with ITSM Rest API Responses 29

4.2.1 HTTP Response Codes 30

4.3 How to call IEM Bi-directional Integration with ITSM APIs 30

4.3.1 API Authentication 30

4.3.2 Calling Integration API with valid Json 34

4.4 IEM Bi-directional Integration with ITSM API Details 37

4.4.1 Generate Token 37

4.4.2 API: SNOW Fetch Ticket Update 38

4.4.3 API: SX Fetch Ticket Update 39

5 **Data View API** 42

5.1 IEM Dataview API Details 42

5.1.1 Generate Token 42

5.1.2 Open actionable API 43

5.1.3 Open Actionable by First Occurrence API 43

5.1.4 Open Actionable by First Occurrence Filtering Between Data APIs 44
5.1.5 Open Actionable by Last Occurrence API 45
5.1.6 Get Related Alerts API 46
5.1.7 Get Related Events API 46
5.1.8 Open Actionable by User API 47
5.1.9 Entity Data API 47
5.1.10 API: AEX integration 48
6 Noise Maintenance Api 50
6.1 Login Details 50
6.1.1 Generate Token 50
6.2 Calling Noise Maintenance APIs 51
6.2.1 Base Filter with Define Maintenance and Without recurring 51
6.2.2 Base Filter and Custom Filter with Define Maintenance and Without recurring 52
6.2.3 Base Filter with Define Maintenance and with Recurring 53
7 API Accessible Matrix 58

Table of Figures

Figure 1 – How to consume token API 13

Figure 2- How to Consume Token API (cont.) 14

Figure 3- How to consume Token API (cont.) 14

Figure 4 - How to consume Token API (cont.) 14

Figure 5 - How to consume Publish APIs 15

Figure 6 - How to consume Publish APIs (cont.) 15

Figure 7 - How to consume Publish APIs (cont.) 15

Figure 8 - How to consume Publish APIs (cont.) 16

Figure 9 - How to consume Publish APIs (cont.) 16

Figure 10 - How to consume Publish APIs (cont.) 16

Figure 11 - How to consume Publish APIs (cont.) 17

Figure 12 - How to consume Publish APIs (cont.) 17

Figure 13 - How to consume Publish APIs (cont.) 17

Figure 14 - How to consume Publish APIs (cont.) 18

Figure 15 - How to consume Publish APIs (cont.) 18

Figure 16 - How to Publish APIs (cont.) 18

Figure 17 - How to consume Publish APIs (cont.) 19

Figure 18 – How to consume token API 30

Figure 19 – How to consume token API (cont.) 31

Figure 20 - How to consume token API (cont.) 31

Figure 21 - How to Consume Token API (cont.) 31

Figure 22 - How to consume Token API (cont.) 32

Figure 23 - How to consume Integration APIs 32

Figure 24 - How to consume Integration APIs (cont.) 32

Figure 25 - How to consume Integration APIs (cont.) 33

Figure 26 - How to consume Integration APIs (cont.) 33

Figure 27 - How to consume Integration APIs (cont.) 33

Figure 28 - How to consume Integration APIs (cont.) 34

Figure 29 - How to consume Integration APIs (cont.) 34

Figure 30 - How to consume Integration APIs (cont.) 34

Figure 31 - How to consume Integration APIs (cont.) 35

Figure 32 - How to consume Integration APIs (cont.) 35

Figure 33 - How to consume Integration APIs (cont.) 35

Figure 34 - How to consume Integration APIs (cont.) 36

Figure 35 - How to consume Integration APIs (cont.) 36

Figure 36 - How to consume Integration APIs (cont.) 36

List of Tables

Table 1 – Conventions 10

Table 2- Request Header Fields 12

Table 3 – HTTP Response Codes 13

Table 4 – IEM Publish Service GET API Details 19

Table 5 – IEM Publish Service POST APIs 19

Table 6 – Generate Token 20

Table 7 – Publish Events Data 21

Table 8 – Publish Metrics Data 22

Table 9 – Publish Entity Topology Data 24

Table 10 – Publish Service Topology Data 25

Table 11 – Publish Healthcheck Data 27

Table 12 - Request Header Fields 29

Table 13 – HTTP Response Codes 30

Table 14 – IEM Bi-directional Integration with ITSM POST APIs 37

Table 15 – Generate Token 37

Table 16 – SNOW Fetch Ticket Update 38

Table 17 – SX Fetch Ticket Update 39

Table 18 – AEX integration Api 48

Table 19 – Generate Token 42

Table 20 – Open actionable Api 43

Table 21 – Open Actionable by Filter Occurrence API 43

Table 22 – Open Actionable by First Occurrence Filtering Between Data APIs 44

Table 23 – Open Actionable by Last Occurrence API 45

Table 24 –Get Related Alerts API 46

Table 25 – Get Related Events API 46

Table 26 – Open Actionable by User API 47

Table 27 – Entity Data API 47

Table 28 – Generate Token 50

Table 29 – Noise Rule creation 51

Table 30 – Noise Rule creation 52

Table 31 – Noise Rule creation ... 53

Table 32 – Noise Rule creation ... 54

Table 33 – Noise Rule creation ... 55

Table 34 - API Accessible Matrix ... 58

Document Revision History

This guide is updated with each release of the product or when necessary.

This table provides the revision history of this API Guide.

Version Date	Description
April, 2024	HCL_IEM_v1.0_API_Guide
January, 2025	HCL_IEM_v1.1_API_Guide
February, 2025	HCL_IEM_v1.2_API_Guide

1 Preface

This section provides information about the IEM PUBLISH SERVICE API Guide, IEM Bi-directional Integration API guide and includes the following topics.

- Intended Audience
- Conventions

1.1 Intended Audience

This guide is intended for users who have access and authorized to use IEM PUBLISH SERVICE, IEM Bi-directional Integration APIs

1.2 Conventions

The following typographic conventions are used in this document:

Table 1 – Conventions

Convention	Element
Boldface	Indicates graphical user interface elements associated with an action, or terms defined in text or the glossary
Blue Underline face	Indicates cross-reference and links
`Courier New (Font)`	Indicates commands within a paragraph, URLs, code in examples, and paths including onscreen text and text input from users
Italic	Indicates document titles, occasional emphasis, or glossary terms
Numbered lists	Indicates steps in a procedure to be followed in a sequence
Bulleted lists	Indicates a list of items that is not necessarily meant to be followed in a sequence

2 IEM API Overview

2.1 IEM Publish Service Overview

IEM Publish is a REST API service which helps to ingest the data into IEM system with different APIs using authentication.

2.2 IEM Bi-directional Integration Overview

IEM Bi-directional Integration is a REST API service which helps to auto update the actionable state to resolve in IEM using authentication when the ticket is getting resolved using the ITSM tool.

3 About IEM Publish Service APIs

IEM Publish service offers a set of REST APIs. Using these APIs, a user establishes the connection between IEM and Nifi connectors using the IMM console, which will help to have continuous live data from Nifi to IEM.

Listed below are the methods that are supported by IEM Publish Service API calls.

- **HTTP POST**: Methods to create or change an object.

3.1 IEM Publish Service Rest API Request

To consume the functionality of IEM, Publish Service provided by a specific API, an enterprise tool needs to place a request. This section details the construction of that request that is supported by IEM Publish Service. Requests are typically categorized in terms of the requested operation: create (POST) or retrieve (GET).

3.1.1 Authentication

HTTP communications between IEM Publish API client and servers are secured with SSL. IEM Publish API supports token-based authentication with cookies. To get a token, the user needs to provide a valid username and password using basic authentication to Token API.

To get the credentials, contact the IEM Admin of an organization (Provider/ organization).

It is important to have the user creds listed below to make a valid API call to IEM Publish API.

- **Email**: A valid username to access the API
- **Password**: A valid password to authenticate the API

3.1.2 Request Header

The following HTTP headers are included in IEM Publish Service API requests.

Table 2- Request Header Fields

Headers	Description
Cookie	All requests from authenticated clients must include “access_token” for cookies in headers

3.2 IEM Publish Service Rest API Responses

Once an enterprise tool requests an IEM Publish service API, an API response is sent to that tool from IEM Publish API post successful authentication. This section details the format of that response.

All responses include a HTTP status code, e.g., 200 for "success", 403 for "Forbidden" etc. Hence, response content depends on the request.

The response body to a POST request contains a message about whether data has been published or not.

3.2.1 HTTP Response Codes

Every response for an IEM Publish Service API call has a response code embedded within it. The table below lists the meaning associated with all the response codes used in IEM Publish Service API call responses. Each of these response codes identifies the reason behind the action that was taken in an API call.

Table 3 – HTTP Response Codes

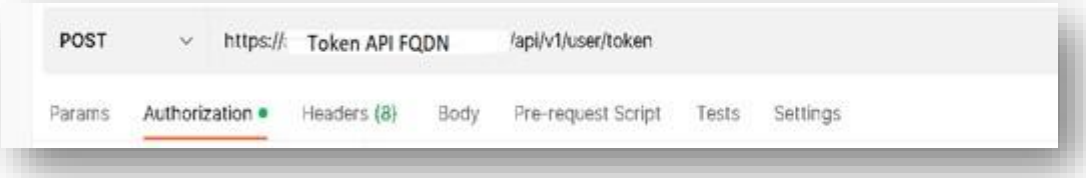
Status Code	Status Description
200 OK	The request is valid and was completed. The response includes a document body.
400 Bad Request	The request body is malformed, incomplete, or otherwise invalid.
401 Unauthorized	An authorization was expected but not found.
403 Forbidden	The response status code indicates that the server understands the request but refuses to authorize it.
404 Not Found	One or more objects specified in the request could not be found in the specified IEM Publish Service API directory.
500 Internal Server Error	The request was received but could not be completed because of an internal error in the server.

3.3 How to call IEM Publish Service APIs

This section explains how users can consume IEM Publish Service APIs. The tool that has been used to demonstrate the IEM Publish Service API consumption in this guide is "**Postman**".

3.3.1 Calling Token API to get a token

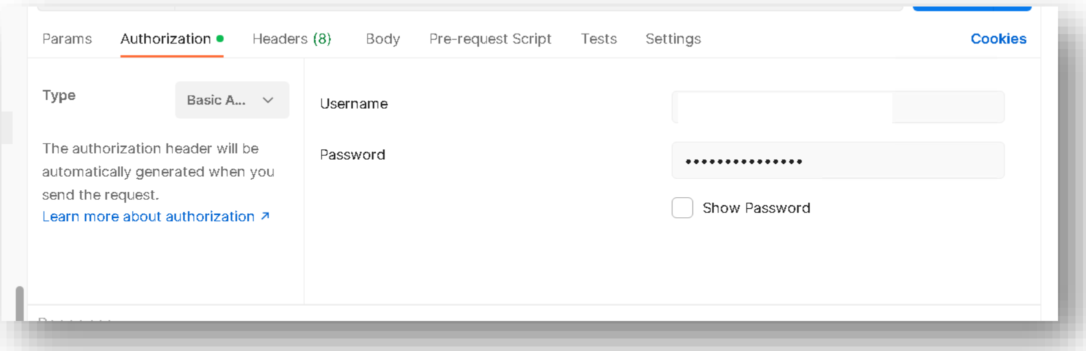
1. Define methodtype = POST



POST https:// Token API FQDN /api/v1/user/token

Figure 1 – How to consume token API

2. Define the basic authentication with username and password.



Type: Basic A...

The authorization header will be automatically generated when you send the request.

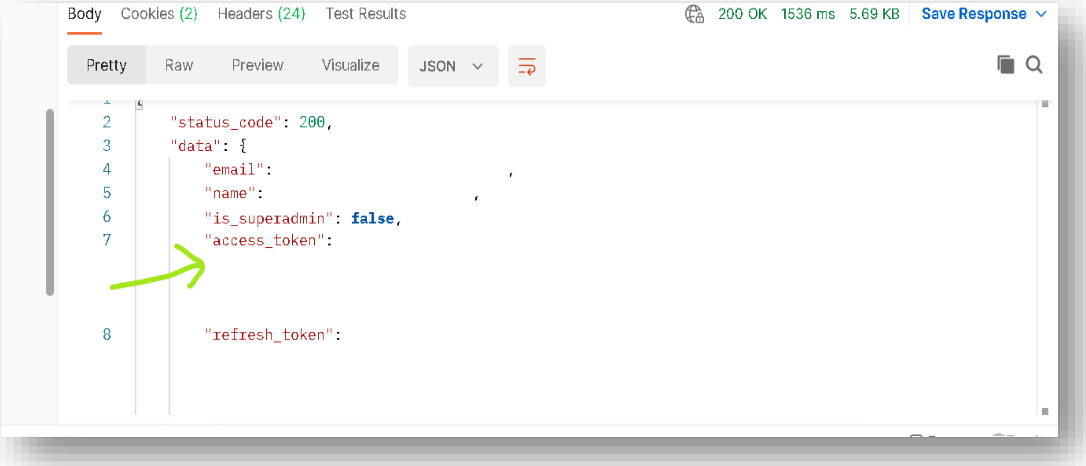
Username

Password: •••••••••••••••

□ Show Password

Figure 2- How to Consume Token API (cont.)

3. Click on the Send button to get a token in response body which will be used to access all the APIs of IEM Publish Service in a session.
   a. If user creds are valid, API will return access token in response.

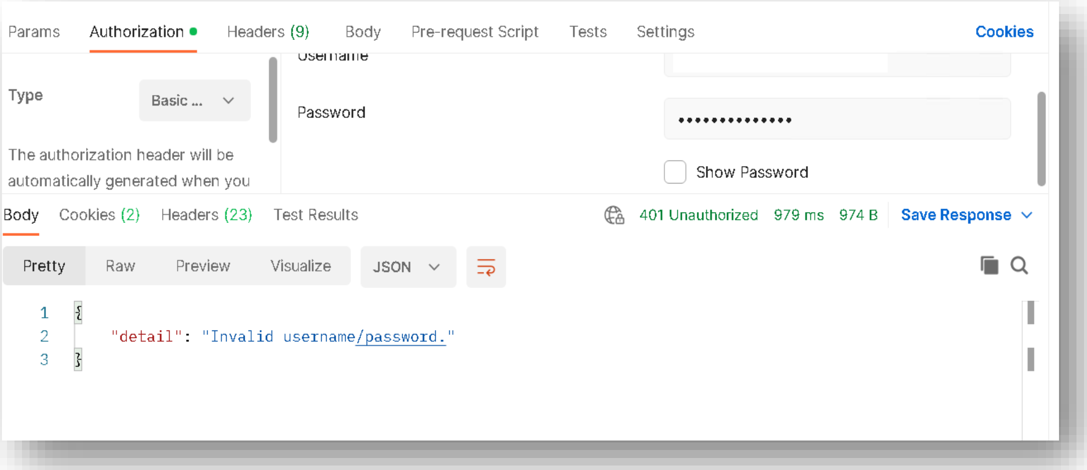


200 OK 1536 ms 5.69 KB

```
{
    "status_code": 200,
    "data": {
        "email":                    ,
        "name":                   ,
        "is_superadmin": false,
        "access_token":


        "refresh_token":
```

Figure 3- How to consume Token API (cont.)

   b. If user creds are invalid, API will return warning message.



Type: Basic ...

The authorization header will be automatically generated when you

Username

Password: ••••••••••••••

□ Show Password

401 Unauthorized 979 ms 974 B

```
{
    "detail": "Invalid username/password."
}
```

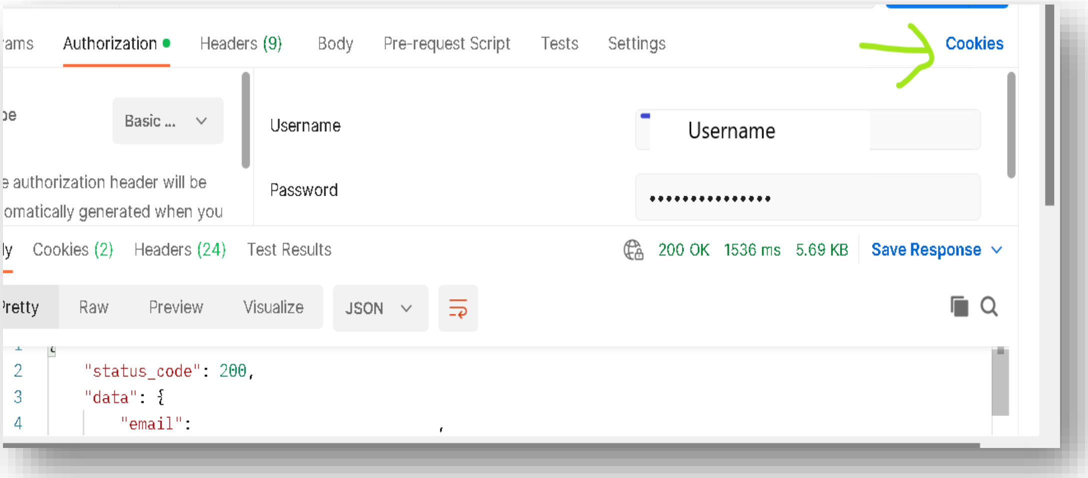
Figure 4 - How to consume Token API (cont.)

3.3.2 Calling Publish API endpoint with access token

1. Set cookies with access token for Publish service API domain

– **Method 1:**

a. Click on cookies.



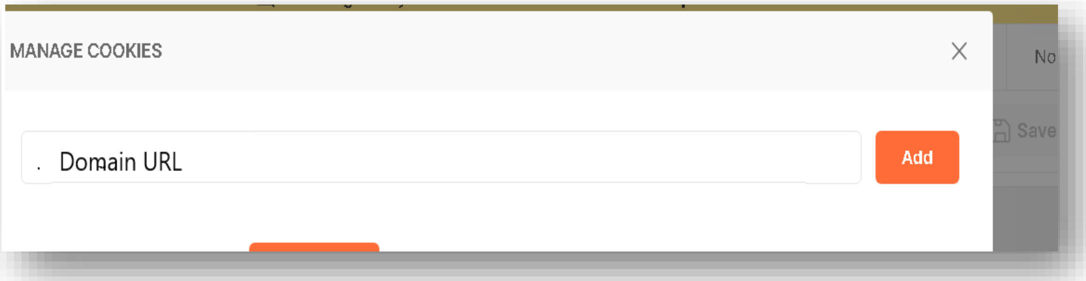
Username: Username

Password: •••••••••••••••

```
{
    "status_code": 200,
    "data": {
        "email":
```

Figure 5 - How to consume Publish APIs

b. Add a domain of Publish Service API



Domain URL

Figure 6 - How to consume Publish APIs (cont.)

c. Click on add cookies.

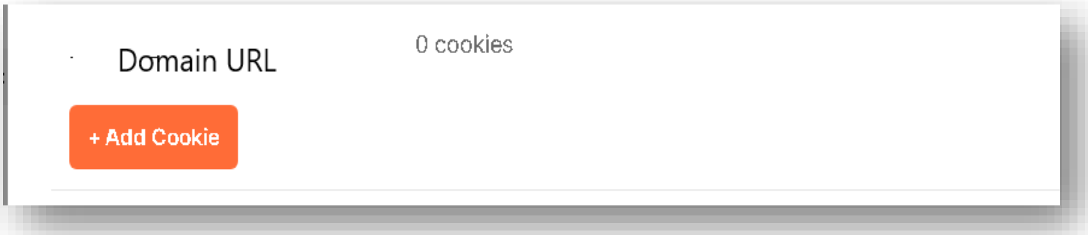
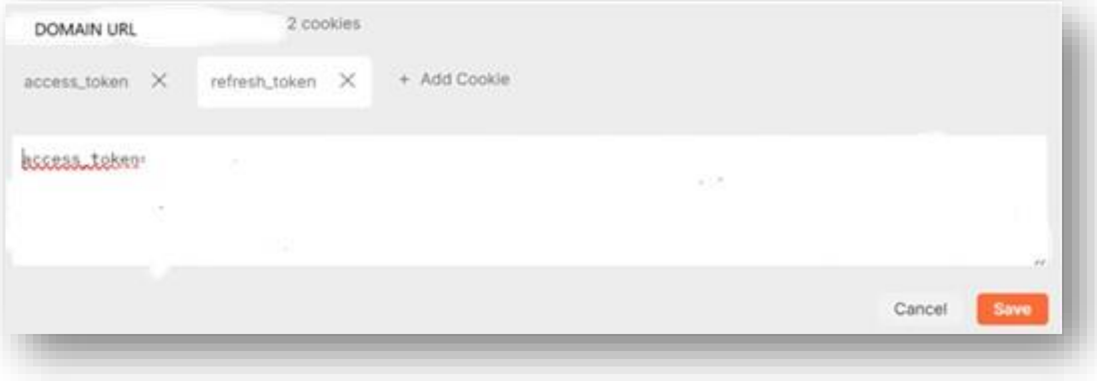


Figure 7 - How to consume Publish APIs (cont.)

d. Set access token key with token value.



access_token=

Figure 8 - How to consume Publish APIs (cont.)

e. Close the cookies window.

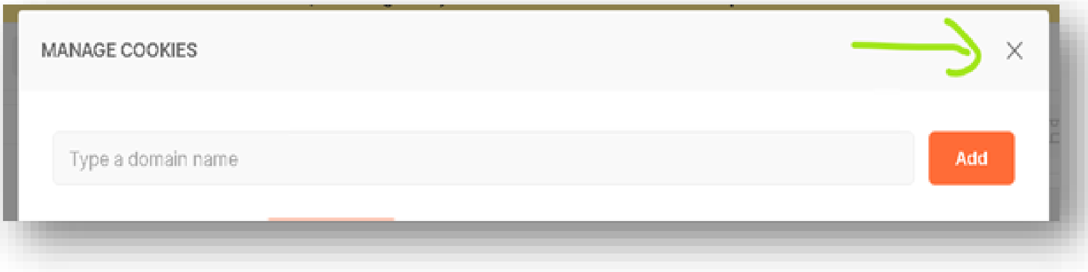
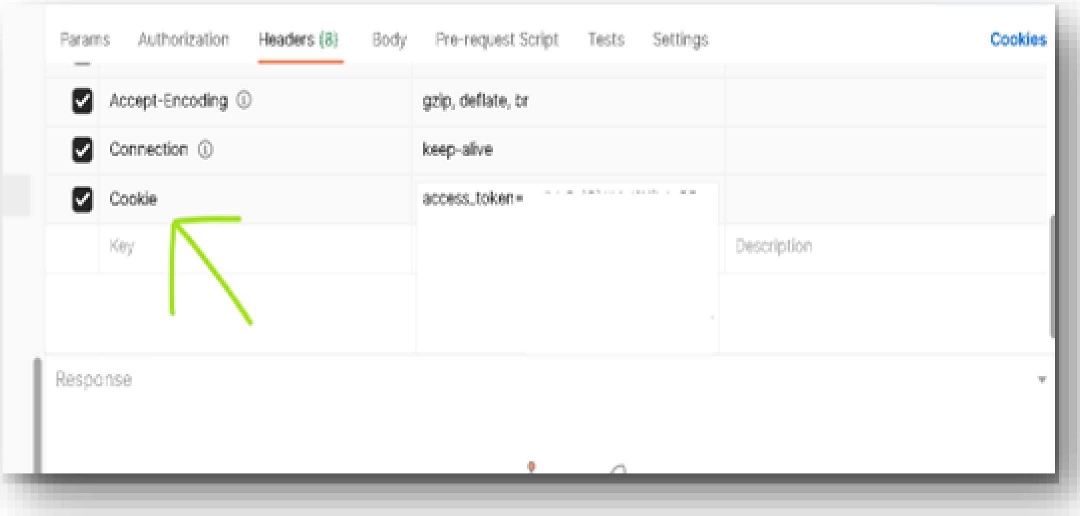


Figure 9 - How to consume Publish APIs (cont.)

– **Method 2:**

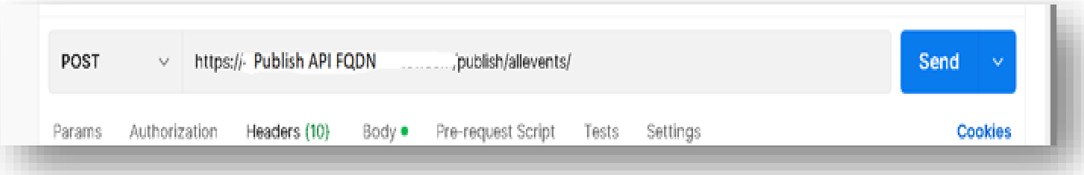
1. Set cookie with access token in API headers



☑	Accept-Encoding ⓘ	gzip, deflate, br
☑	Connection ⓘ	keep-alive
☑	Cookie	access_token=
	Key		Description

Figure 10 - How to consume Publish APIs (cont.)

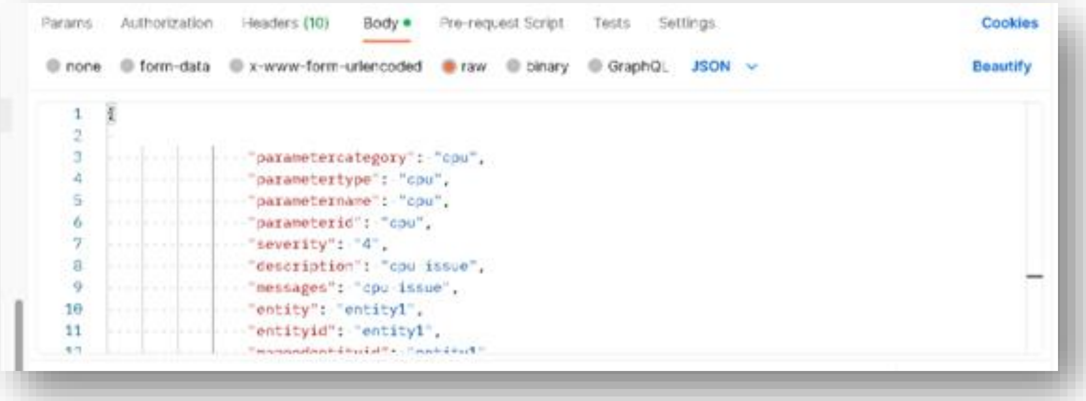
2. Enter the Publish API endpoint with “POST” as method type



POST https://- Publish API FQDN ..../publish/allevents/

Figure 11 - How to consume Publish APIs (cont.)

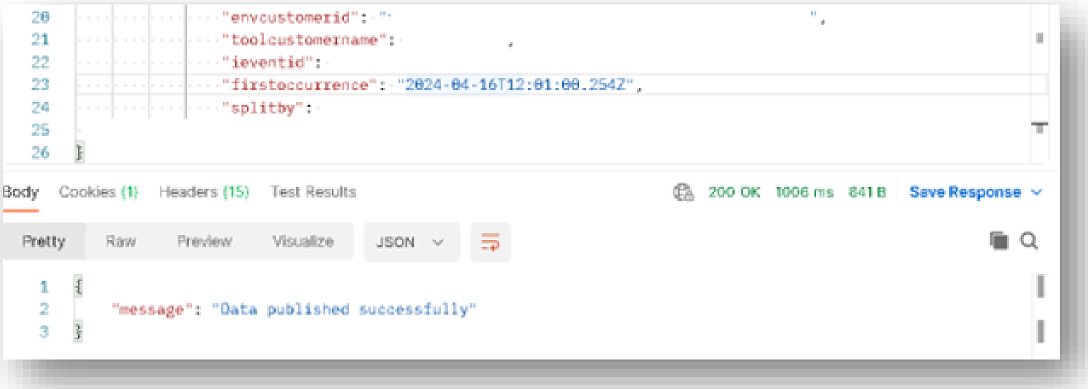
3. Enter valid Json for API body/payload



```
{

            "parametercategory": "cpu",
            "parametertype": "cpu",
            "parametername": "cpu",
            "parameterid": "cpu",
            "severity": "4",
            "description": "cpu issue",
            "messages": "cpu issue",
            "entity": "entity1",
            "entityid": "entity1",
```

Figure 12 - How to consume Publish APIs (cont.)

4. Click on Send to get API response.
5. API response will contain a success or failure message based on Json provided
   a. Success case



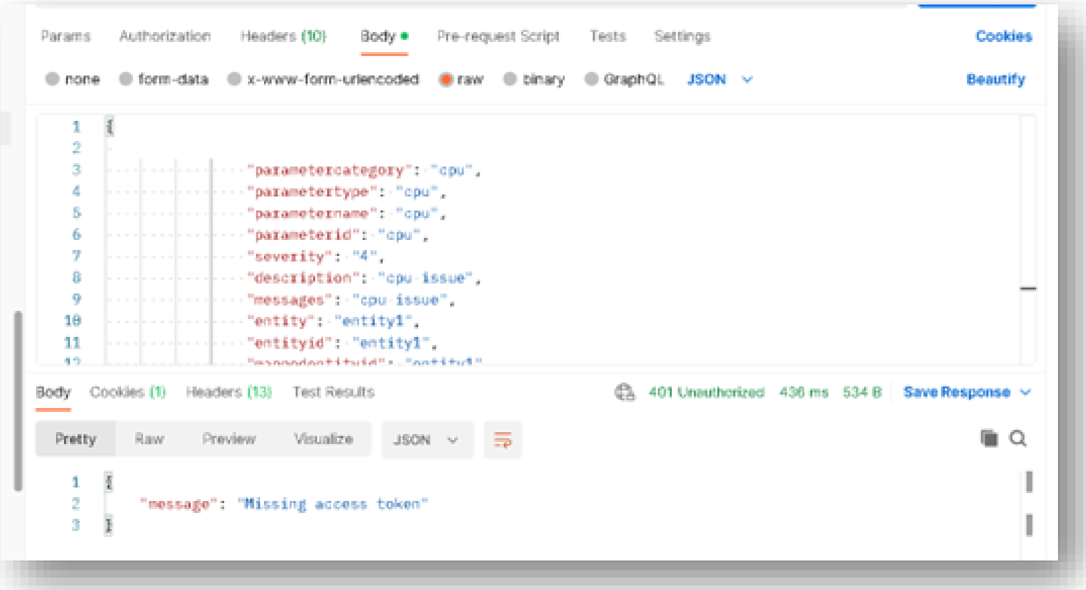
```
            "envcustomerid": "                    ",
            "toolcustomername":           ,
            "ieventid":
            "firstoccurrence": "2024-04-16T12:01:00.254Z",
            "splitby":

}
```

200 OK 1006 ms 841 B

```
{
    "message": "Data published successfully"
}
```

Figure 13 - How to consume Publish APIs (cont.)

   b. Failure cases
      i. When access token is not provided

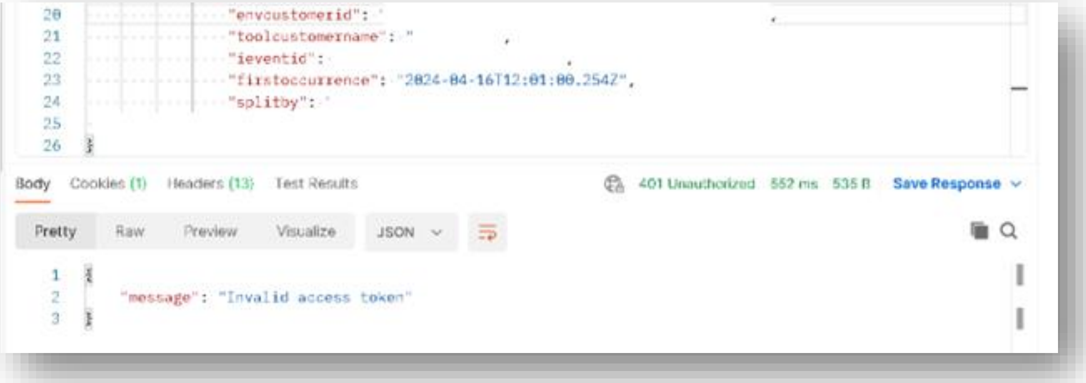


```
{

            "parametercategory": "cpu",
            "parametertype": "cpu",
            "parametername": "cpu",
            "parameterid": "cpu",
            "severity": "4",
            "description": "cpu issue",
            "messages": "cpu issue",
            "entity": "entity1",
            "entityid": "entity1",
```

```
{
    "message": "Missing access token"
}
```

Figure 14 - How to consume Publish APIs (cont.)

ii. When invalid token value is provided

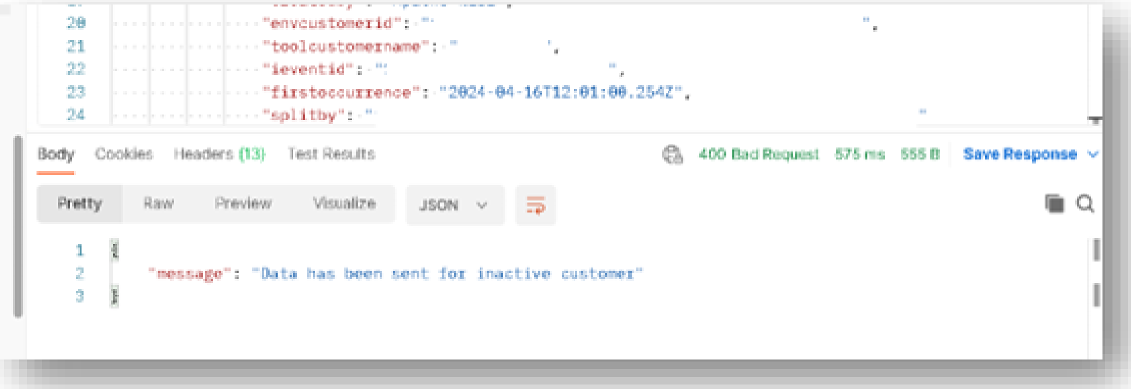


```
            "envcustomerid": "
            "toolcustomername": "
            "ieventid":
            "firstoccurrence": "2024-04-16T12:01:00.254Z",
            "splitby": "

}
```

```
{
    "message": "Invalid access token"
}
```

Figure 15 - How to consume Publish APIs (cont.)

iii. When data has been sent for inactive customer or customerid is invalid

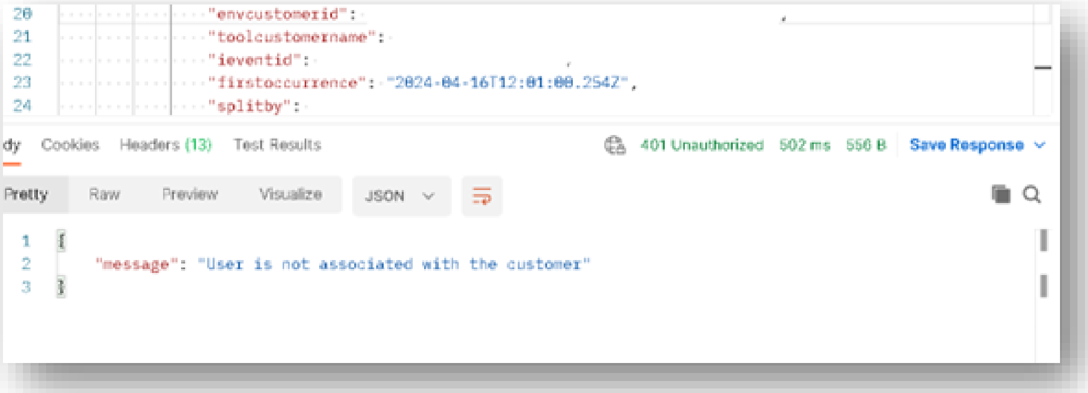


```
            "envcustomerid": "                    ",
            "toolcustomername": "          ',
            "ieventid": "             ",
            "firstoccurrence": "2024-04-16T12:01:00.254Z",
            "splitby": "                             "
```

```
{
    "message": "Data has been sent for inactive customer"
}
```

Figure 16 - How to Publish APIs (cont.)

iv. When customer id is valid, but user is not mapped to that customer



```
"envcustomerid":
"toolcustomername":
"ieventid":
"firstoccurrence": "2024-04-16T12:01:00.254Z",
"splitby":
```

```
{
    "message": "User is not associated with the customer"
}
```

Figure 17 - How to consume Publish APIs (cont.)

3.4 IEM Publish Service API Details

This section provides the details of all the existing APIs for IEM Publish Service. The APIs in the current version of IEM Publish Service utilize the GET and POST methods. Listed in the table below are the APIs categorized based on these methods.

Table 4 – IEM Publish Service GET API Details

POST APIs – To publish data into IEM

S. No.	API	API Description
1	https: {{domain}}/publish/health check/	The API returns a message "API is running" with 200 status code

Table 5 – IEM Publish Service POST APIs

POST APIs – To publish data into IEM

S. No.	API	API Description
1	Generate Token	This API is used to generate the security token with the valid user credentials.
2	Publish Events data	This API is to send the events data into GCP pubsublite topics, it will be processed further and will be visible over IEM console
3	Publish Metrics data	This API is to send the metrics data into GCP pubsublite topics, it will be processed further and will be visible over IEM console
4	Publish Entity Topology data	This API is to send the entity topology data into GCP pubsublite topics, it will be processed further and will be visible over IEM console
5	Publish Service Topology data	This API is to send the service topology data into GCP pubsublite topics, it will be processed further and will be visible over IEM console

6	Publish Healthcheck data	This API is to send the Nifi adaptors health check data into GCP pubsublite topics, it will be processed further and will be visible over IEM console

3.4.1 Generate Token

Table 6 – Generate Token

Element	Description
API	Generate Token
Description	API returns the access token based on username and password to authenticate IEM Publish APIs.
Method	POST
URL	https://<Hosted_API>/api/v1/user/token
Body	NA
Header	Authorization headers using basic authentication with username and password

Response (Success Case: 200 OK status code)	{
	    "status_code": 200,
	    "data": {
	        "email": "<user_email>",
	        "name": "<User name>",
	        "is_superadmin": false,
	        "access_token": "<token_value>",
	        "refresh_token": "<refresh_token_value>",
	        "associated_with_customers": [
	            {}
	
	        ]
	    },
	    "msg": ""
	}

Response Parameter	Access_token

Response (Failed Case: 401 Unauthorized status code)	{
	"detail": "Invalid
	username/password."
	}

3.4.2 API: Publish Events Data

Table 7 – Publish Events Data

Element	Description
API	Publish Events data
Description	This API is to send the events data into GCP pubsublite topics, it will be processed further and will be visible over IEM console
Method	POST
URL	https://<Hosted_API>/publish/allevents/
Header	Cookies with access_token value
Body	{ "parametercategory": "cpu", "parametertype": "cpu", "parametername": "cpu", "parameterid": "cpu", "severity": "4", "description": "cpu issue", "messages": "cpu issue", "entity": "entity1", "entityid": "entity1", "mappedentityid": "entity1", "agentlocation": "<agentlocation_value>", "createdon": "2024-04- 16T12:01:00.254Z", "toolmanager": "<toolmanager>", "manageragent": "<manager_agent>", "datasourcename": "<datasource>", "toolcustomerid": "<customer_name>",

	"createdby": "", "envcustomerid": "<customer_id>", "toolcustomername": "<customer_name>", "ieventid": "11-j2-5-c-asd-nd34b46644", "firstoccurrence": "2024-04-16T12:01:00.254Z", "splitby": "<customer_id>_entity1_cpu" }

Response
(Success Case: 200 OK status code)
	{
	"message": "Data published successfully"
	}

Response
(Failed Case 1: 401 Unauthorized status code)
	{
	"message": "Missing access token"
	}

Response
(Failed Case 2: 401 Unauthorized status code)
	{
	"message": "Invalid access token"
	}

Response
(Failed Case 3: 401 Unauthorized status code)
	{
	"message": "Access token has expired"
	}

Response
(Failed Case 4: 401 Unauthorized status code)
	{
	"message": "Data has been sent for inactive customer"
	}

Response
(Failed Case 5: 401 Unauthorized status code)
	{
	"message": "User is not associated with the customer"
	}

3.4.3 API: Publish Metrics Data

Table 8 – Publish Metrics Data

Element	Description
API	Publish Metrics data
Description	This API is to send the metrics data into GCP pubsublite topics, it will be processed further and will be visible over IEM console
Method	POST
URL	https://<Hosted_API>/publish/metrics/
Header	Cookies with access_token value

Body	{
	"parametercategory": "cpu",
	"parametertype": "cpu",
	"parametername": "cpu",
	"parameterid": "cpu",
	"severity": "4",
	"description": "cpu issue",
	"messages": "cpu issue",
	"entity": "entity1",
	"entityid": "entity1",
	"mappedentityid": "entity1",
	"agentlocation": "<agentlocation_value>",
	"createdon": "2024-04-16T12:01:00.254Z",
	"toolmanager": "<toolmanager>",
	"manageragent": "<manager_agent>",
	"datasourcename": "<datasource>",
	"toolcustomerid": "<customer_name>",
	"createdby": "",
	"envcustomerid": "<customer_id>",
	"toolcustomername": "<customer_name>",
	"ieventid": "11-j2-5-c-asd-nd34b46644",
	"firstoccurrence": "2024-04-16T12:01:00.254Z",
	"splitby": "<customer_id>_entity1_cpu"
	}

Response	{ "message": "Data published successfully"

(Success Case: 200 OK status code)	`}`

Response
(Failed Case 1: 401 Unauthorized status code)
	`{`
	`    "message": "Missing access token"`
	`}`

Response
(Failed Case 2: 401 Unauthorized status code)
	`{`
	`    "message": "Invalid access token"`
	`}`

Response
(Failed Case 3: 401 Unauthorized status code)
	`{`
	`    "message": "Access token has expired"`
	`}`

Response
(Failed Case 4: 401 Unauthorized status code)
	`{`
	`    "message": "Data has been sent for inactive customer"`
	`}`

Response
(Failed Case 5: 401 Unauthorized status code)
	`{`
	`    "message": "User is not associated with the customer"`
	`}`

3.4.4 API: Publish Entity Topology Data

Table 9 – Publish Entity Topology Data

Element	Description
API	Publish Entity Topology data
Description	This API is to send the entity topology data into GCP pubsublite topics, it will be processed further and will be visible over IEM console
Method	POST
URL	https://<Hosted_API>/publish/entity/
Header	Cookies with access_token value

Body	`{`
	`  "custcol”: {`

	"Category”: "Business Application",
	"Location”: "India"
	},
	"entityid”: "testentityj1",
	"topology”: {
	"parententity”: "testentityj1"
	},
	"envcustomer_id”: "<customer_id>",
	"createdon”: "2024-01-23T11:02:59.315Z"
	}

Response
(Success Case: 200 OK status code)
	{
	"message": "Data published successfully"
	}

Response
(Failed Case 1: 401 Unauthorized status code)
	{
	"message": "Missing access token"
	}

Response
(Failed Case 2: 401 Unauthorized status code)
	{
	"message": "Invalid access token"
	}

Response
(Failed Case 3: 401 Unauthorized status code)
	{
	"message": "Access token has expired"
	}

Response
(Failed Case 4: 401 Unauthorized status code)
	{
	"message": "Data has been sent for inactive customer"
	}

Response
(Failed Case 5: 401 Unauthorized status code)
	{
	"message": "User is not associated with the customer"
	}

3.4.5 API: Publish Service Topology Data

Table 10 – Publish Service Topology Data

Element	Description

API	Publish Service Topology data
Description	This API is to send the service topology data into GCP pubsublite topics, It will be processed further and will be visible over IEM console
Method	POST
URL	https://<Hosted_API>/publish/service/
Header	Cookies with access_token value

Body	{
	"servicename":"testservicej1",
	"service_entity_mapping":[
	{"entityid":"testentityj1"},
	{"entityid":"testentityj2"}],
	"service_topology":[
	{"parentservicename":"testservicej2"},
	{"parentservicename":"testservicej3"}
	],
	"envcustomer_id":"<customer_id>"
	}

Response
(Success Case: 200 OK status code)
	{
	"message": "Data published successfully"
	}

Response
(Failed Case 1: 401 Unauthorized status code)
	{
	"message": "Missing access token"
	}

Response
(Failed Case 2: 401 Unauthorized status code)
	{
	"message": "Invalid access token"
	}

Response
(Failed Case 3: 401 Unauthorized status code)
	{
	"message": "Access token has expired"
	}

Response	{ "message": "Data has been sent for inactive customer"

(Failed Case 4: 401 Unauthorized status code)	}

Response
(Failed Case 5: 401 Unauthorized status code)
	{
	"message": "User is not associated with the customer"
	}

3.4.6 API: Publish Healthcheck Data

Table 11 – Publish Healthcheck Data

Element	Description
API	Publish Healthcheck data
Description	This API is to send the Nifi adaptors healthcheck data into GCP pubsublite topics, it will be processed further and will be visible over IEM console
Method	POST
URL	https://<Hosted_API>/publish/healthcheck/
Header	Cookies with access_token value

Body	{
	"host":"test host2",
	"envcustomer_id": "<customer_id>",
	"kpi_name":"heartbeat",
	"kpi_type":"heartbeat",
	"status":"Green",
	"value":"",
	"description":"Heartbeat received on 2024-02-19 10:20:00.477",
	"remark":"Heartbeat received on 2024-02-19 10:20:00.477",
	"timestamp":"2024-02-19 10:20:00.477",
	"updatetimestamp":"2024-02-19 10:20:00.477",
	"updateid":"Manual testing",
	"additional_info": "{}"
	}

Response
(Success Case: 200 OK status code)
	{
	"message": "Data published successfully"
	}

Response
(Failed Case 1: 401 Unauthorized status code)
	{
	    "message": "Missing access token"
	}

Response
(Failed Case 2: 401 Unauthorized status code)
	{
	    "message": "Invalid access token"
	}

Response
(Failed Case 3: 401 Unauthorized status code)
	{
	    "message": "Access token has expired"
	}

Response
(Failed Case 4: 401 Unauthorized status code)
	{
	    "message": "Data has been sent for inactive customer"
	}

Response
(Failed Case 5: 401 Unauthorized status code)
	{
	    "message": "User is not associated with the customer"
	}

4 About IEM Bi-directional Integration with ITSM APIs

IEM Bi-directional Integration with ITSM offers a set of REST APIs. Using these APIs, a user establishes the connection between IEM and ITSM tools (ex: SNOW, Service Exchange), which will help to auto update the actionable state in IEM when resolving the ticket at ITSM tool.

Listed below are the methods that are supported by IEM Bi-directional Integration with ITSM API calls

- **HTTP POST:** Methods to create or change an object.

4.1 IEM Bi-directional Integration with ITSM Rest API Request

To consume the functionality of IEM Bi-directional Integration with ITSM provided by a specific API, an enterprise tool needs to place a request. This section details the construction of that request that is supported by IEM Bi-directional Integration with ITSM. Requests are typically categorized in terms of the requested operation: create (POST).

4.1.1 Authentication

HTTP communications between IEM Integration API clients and servers are secured with SSL. IEM Bi-directional Integration API supports.

1. Token-based authentication with cookies
2. Basic authentication using username and password.

To get a token, the user needs to provide a valid username and password using basic authentication to take API.

To get the credentials, contact the IEM Admin of an organization (Provider/ organization).

It is important to have the user creds listed below to make a valid API call to IEM Bi-directional Integration API.

- **Email**: A valid username to access the API
- **Password**: A valid password to authenticate the API

4.1.2 Request Header

The following HTTP headers are included in IEM Bi-directional Integration API requests.

Table 12 - Request Header Fields

Headers	Description
Cookie	All requests from authenticated clients must include “access_token” for cookies in headers
Authorization	Basic authentication header using username and password

4.2 IEM Bi-directional Integration with ITSM Rest API Responses

Once an enterprise tool requests an IEM Bi-directional Integration API, an API response is sent to that tool from IEM Integration API post successful authentication. This section details the format of that response.

All responses include a HTTP status code, e.g., 200 for “success”, 403 for “Forbidden” etc. Hence, response content depends on the request.

The response body to a POST request contains a message if the request is successful or not.

4.2.1 HTTP Response Codes

Every response for an IEM Bi-directional Integration API call has a response code embedded within it. The table below lists the meaning associated with all the response codes used in IEM Bi-directional Integration API call responses. Each of these response codes identifies the reason behind the action that was taken in an API call.

Table 13 – HTTP Response Codes

Status Code	Status Description
200 OK	The request is valid and was completed. The response includes a document body.
400 BadRequest	The request body is malformed, incomplete, or otherwise invalid.
401 Unauthorized	An authorization header was expected but not found.
403 Forbidden	The response status code indicates that the server understands the request but refuses to authorize it.
404 Not Found	One or more objects specified in the request could not be found in specified IEM Bi-directional Integration with ITSM API directory.
500 Internal Server Error	The request was received but could not be completed because of an internal error in the server.

4.3 How to call IEM Bi-directional Integration with ITSM APIs

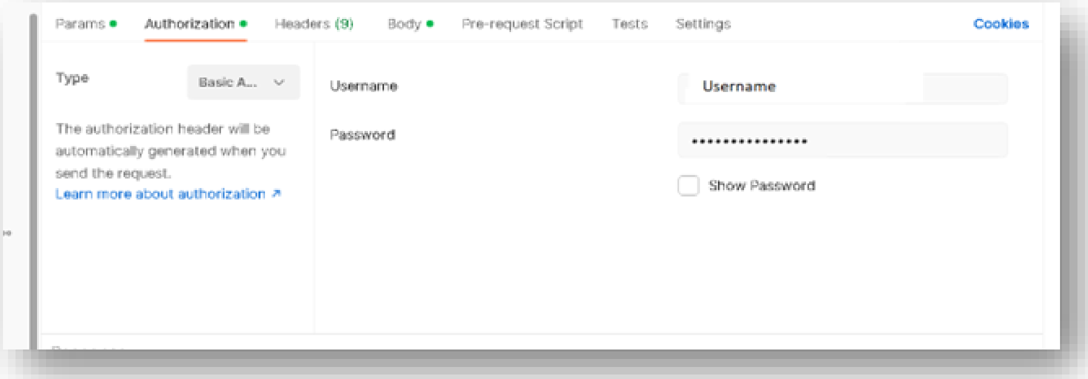
This section explains how users can consume IEM Bi-directional Integration APIs. In this guide “Postman” tool has been used to explain IEM Integration API calls and considering “SNOW” as ITSM tool.

4.3.1 API Authentication

Bi-directional Integration API supports two types of authentications as follows

1. Basic authentication using username and password.

– Provide username and password for authorization using basic authentication to Integration API.



Type: Basic A...

The authorization header will be automatically generated when you send the request.

Learn more about authorization ↗

Username: Username

Password: •••••••••••••••

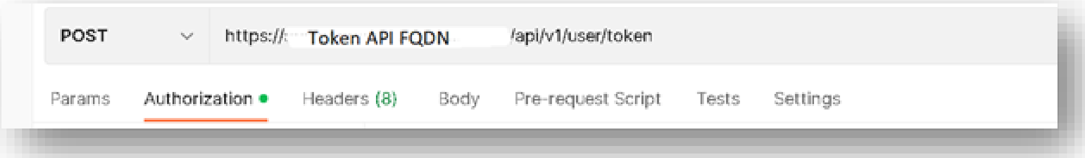
□ Show Password

Figure 18 – How to consume token API

2. Token based authentication

– **Calling Token API to get a token**

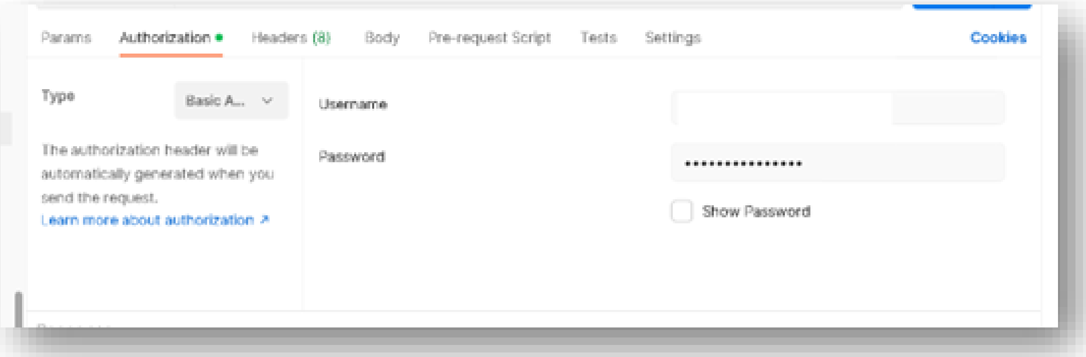
a. Define methodtype = POST



POST https://Token API FQDN/api/v1/user/token

Figure 19 – How to consume token API (cont.)

b. Define the basic authentication with username and password to token API



Type: Basic A...

The authorization header will be automatically generated when you send the request.

Learn more about authorization ↗

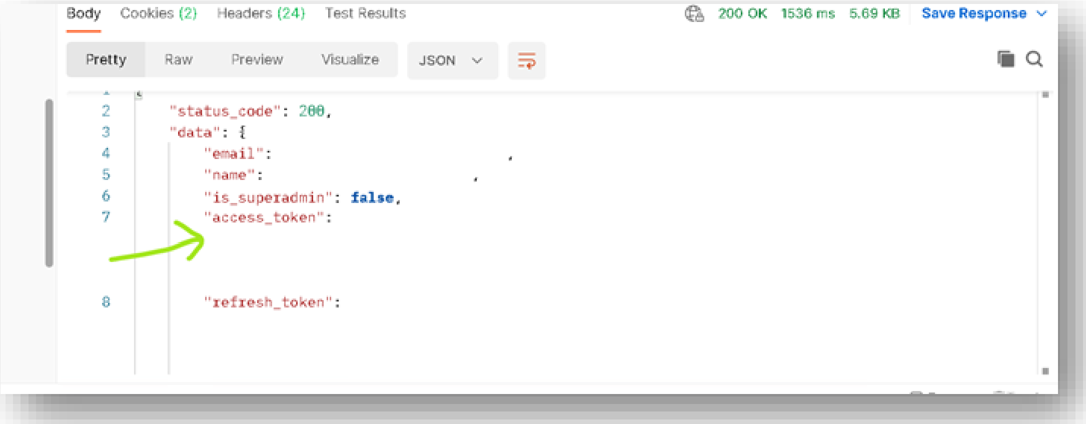
Username

Password: ••••••••••••••••

□ Show Password

Figure 20 - How to consume token API (cont.)

3. Click on the Send button to get a token in response body which will be used to access the APIs of IEM Bi-directional Integration in a session.
   a. If user creds are valid, API will return access token in response.

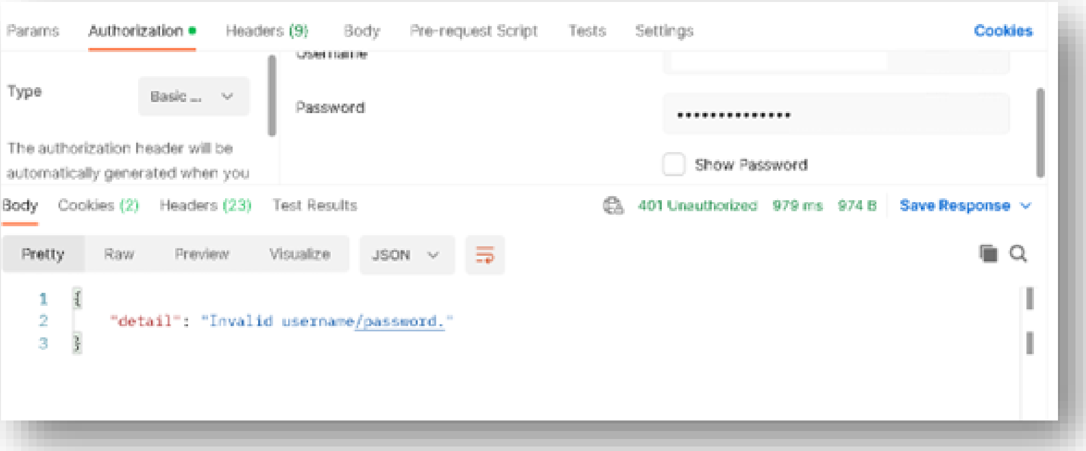


200 OK 1536 ms 5.69 KB

```
    "status_code": 200,
    "data": {
        "email":
        "name":
        "is_superadmin": false,
        "access_token":

        "refresh_token":
```

Figure 21 - How to Consume Token API (cont.)

b. If user creds are invalid, token API will return a message.



Type: Basic ...

The authorization header will be automatically generated when you

Username

Password: •••••••••••••••

□ Show Password

401 Unauthorized 979 ms 974 B

JSON

```
{
    "detail": "Invalid username/password."
}
```

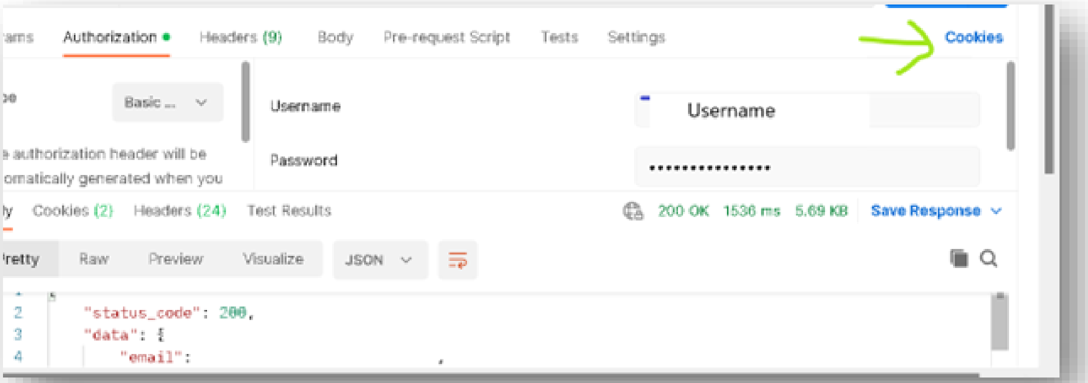
Figure 22 - How to consume Token API (cont.)

- **Calling Integration API endpoint with access token**

1. Set cookies with access token for Bi-directional Integration API domain

- **Method 1:**

a. Click on cookies.



pe: Basic ...

e authorization header will be omatically generated when you

Username: Username

Password: •••••••••••••••

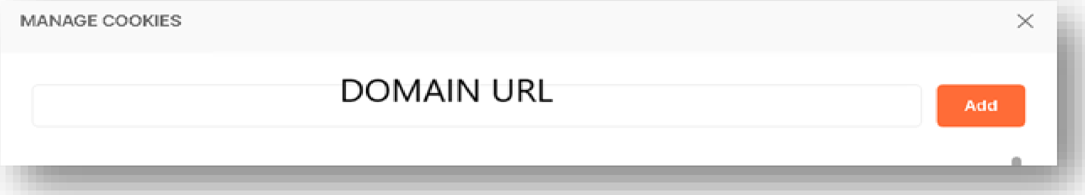
200 OK 1536 ms 5.69 KB

JSON

```
    "status_code": 200,
    "data": {
        "email":
```

Figure 23 - How to consume Integration APIs

b. Add a domain of Bi-directional Integration with ITSM API



DOMAIN URL

Figure 24 - How to consume Integration APIs (cont.)

c. Click on add cookies.

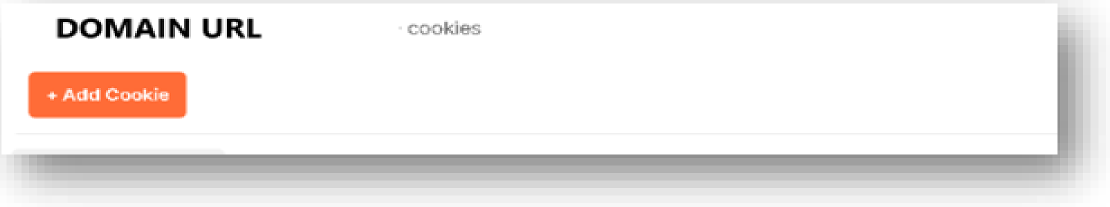


Figure 25 - How to consume Integration APIs (cont.)

d. Set access token key with token value.

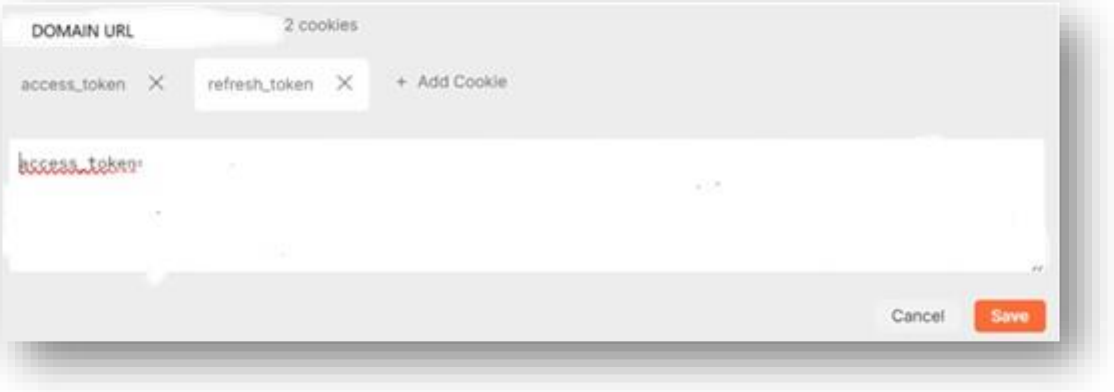


Figure 26 - How to consume Integration APIs (cont.)

e. Close the cookies window.

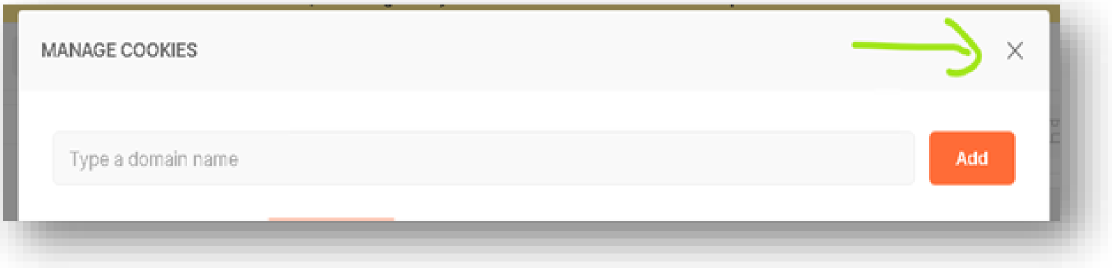
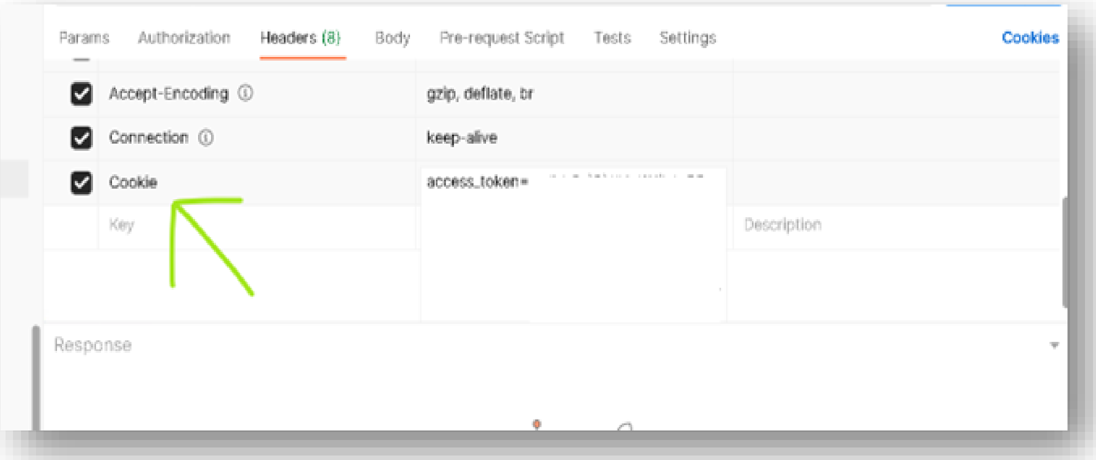


Figure 27 - How to consume Integration APIs (cont.)

- **Method 2:**

a. Set cookie with access token in Integration API headers.

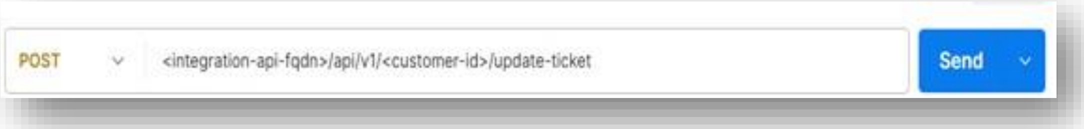


☑ Accept-Encoding ⓘ	gzip, deflate, br	
☑ Connection ⓘ	keep-alive	
☑ Cookie	access_token=	
Key		Description

Figure 28 - How to consume Integration APIs (cont.)

4.3.2 Calling Integration API with valid Json

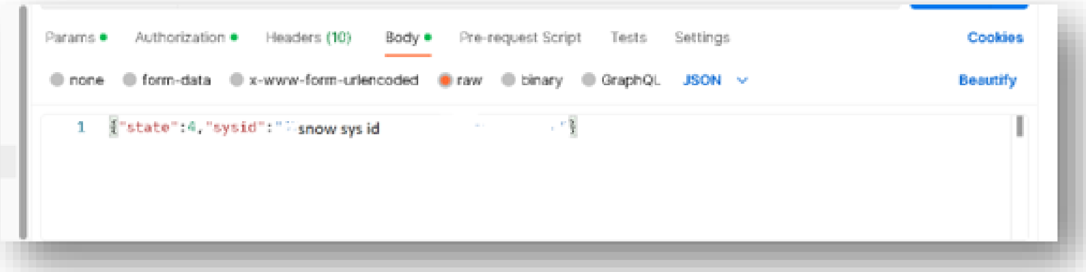
1. Enter the Integration API endpoint with “POST” as method type.



POST <integration-api-fqdn>/api/v1/<customer-id>/update-ticket

Figure 29 - How to consume Integration APIs (cont.)

2. Enter valid Json for API body/payload

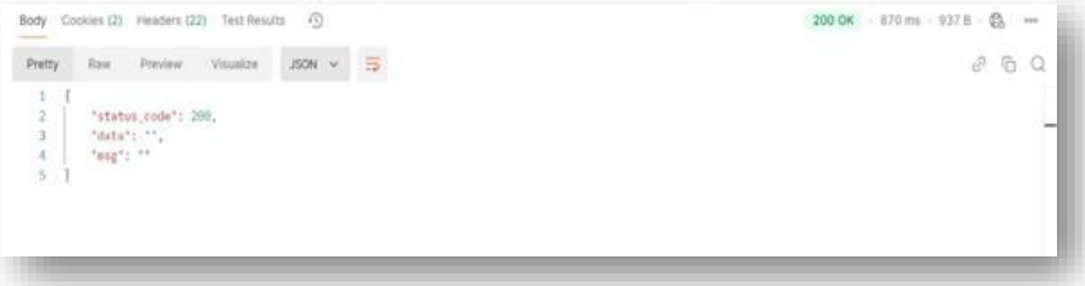


○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{"state":4,"sysid":" snow sys id "}
```

Figure 30 - How to consume Integration APIs (cont.)

3. Click on Send to get API response.
4. API response will contain a success or failure message based on Json provided
   a. Success case

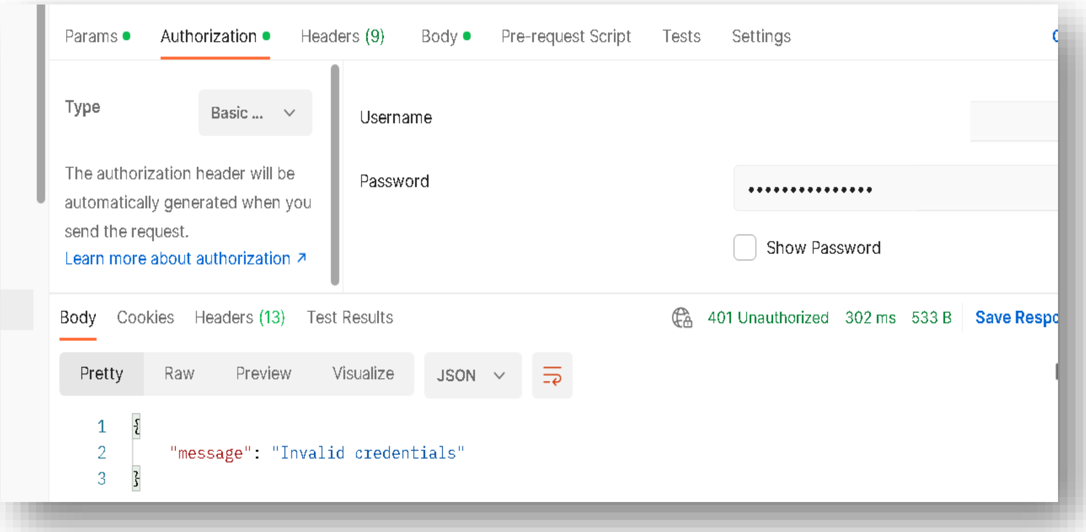


```
{
    "status_code": 200,
    "data": "",
    "msg": ""
}
```

Figure 31 - How to consume Integration APIs (cont.)

b. Failure cases

i. When user creds are invalid by using basic authentication



Type: Basic ...

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Username

Password: •••••••••••••••

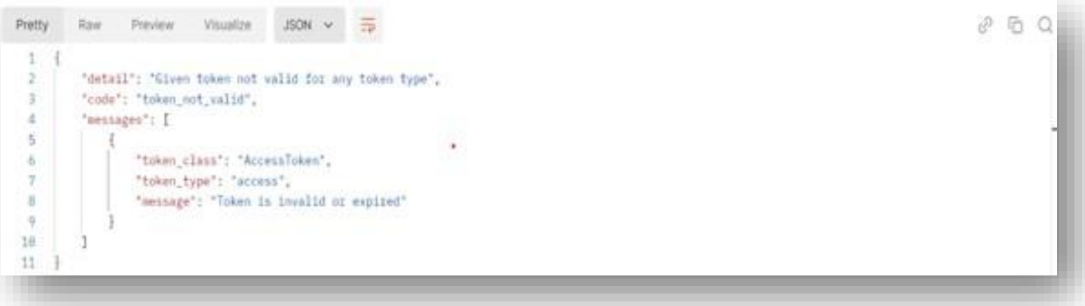
□ Show Password

401 Unauthorized 302 ms 533 B

```
{
    "message": "Invalid credentials"
}
```

Figure 32 - How to consume Integration APIs (cont.)

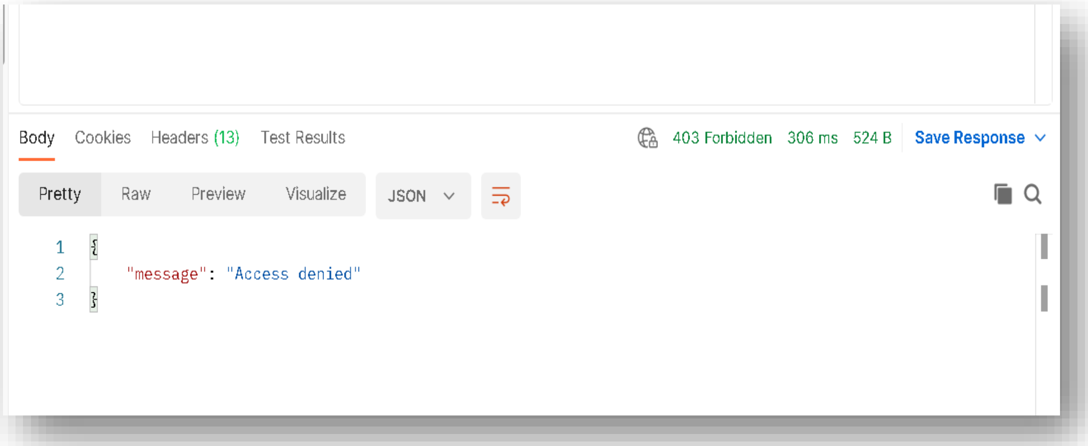
ii. When invalid token value is provided using token-based authentication



```
{
    "detail": "Given token not valid for any token type",
    "code": "token_not_valid",
    "messages": [
        {
            "token_class": "AccessToken",
            "token_type": "access",
            "message": "Token is invalid or expired"
        }
    ]
}
```

Figure 33 - How to consume Integration APIs (cont.)

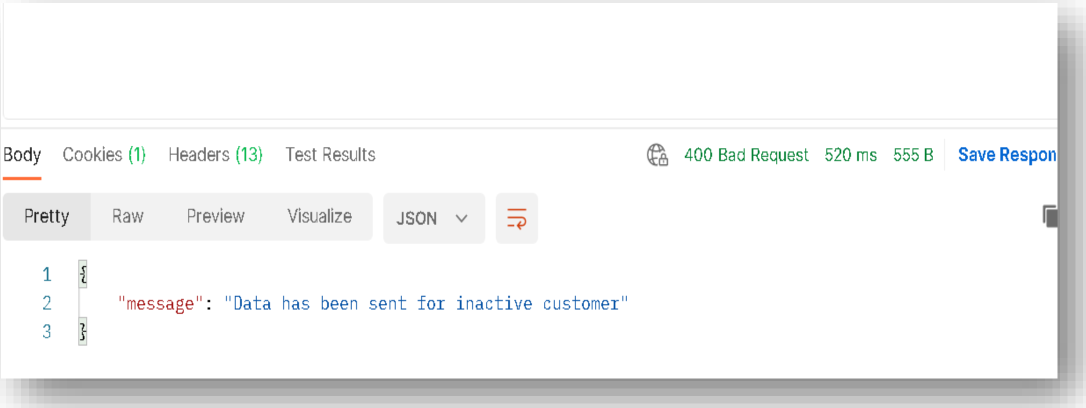
iii. When authentication is not done with the integration API



```
{
    "message": "Access denied"
}
```

Figure 34 - How to consume Integration APIs (cont.)

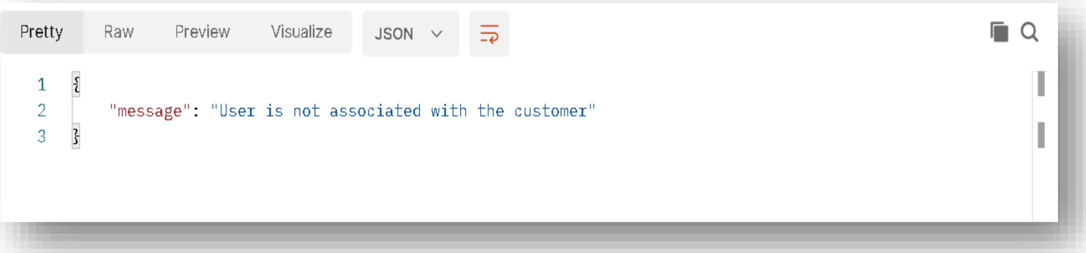
iv. When invalid customer id provided in URL parameters



```
{
    "message": "Data has been sent for inactive customer"
}
```

Figure 35 - How to consume Integration APIs (cont.)

v. When user not associated with the customer provided in URL parameters



```
{
    "message": "User is not associated with the customer"
}
```

Figure 36 - How to consume Integration APIs (cont.)

4.4 IEM Bi-directional Integration with ITSM API Details

This section provides the details of all the existing APIs for IEM Bi-directional Integration with ITSM. The APIs in the current version of IEM Bi-directional Integration with ITSM have POST methods.

Table 14 – IEM Bi-directional Integration with ITSM POST APIs

POST APIS – To publish data into IEM		
S. No.	**API**	**API Description**
1	Generate Token	This API is used to generate security token with the valid user credentials.
2	Fetch Ticket Update	This API is to update the actionable state in IEM whenever ticket is getting resolved at ITSM tool

4.4.1 Generate Token

Table 15 – Generate Token

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>API</td><td>Generate Token</td></tr>
<tr><td>Description</td><td>API returns the access token based on username and password to authenticate IEM Integration APIs.</td></tr>
<tr><td>Method</td><td>POST</td></tr>
<tr><td>URL</td><td>https://<Hosted_API>/api/v1/user/token</td></tr>
<tr><td>Body</td><td>NA</td></tr>
<tr><td>Header</td><td>Authorization headers using basic authentication with username and password</td></tr>
<tr><td>Response (Success Case: 200 OK status code)</td><td><pre>
{
    "status_code": 200,
    "data": {
        "email": "<user_email_id>",
        "name": "<user_name>",
        "is_superadmin": false,
        "access_token": "<access_token_value>",
        "refresh_token": "<refresh_token_value>",
        "associated_with_customers": [
            {}
                            ]
    },
    "msg": ""
}
</pre></td></tr>
<tr><td>Response Parameter</td><td>Access_token</td></tr>
</table>

Response (Failed Case: 401 Unauthorized status code)	{
	    "detail": "Invalid username/password."
	}

4.4.2 API: SNOW Fetch Ticket Update

Table 16 – SNOW Fetch Ticket Update

Element	Description
API	Fetch Ticket Update
Description	This API is to update the actionable state in IEM whenever ticket is getting resolved at ITSM tool.
Method	POST
URL	https://<Hosted_API>/<customerid>/update-ticket?integrationid=<integrationid>
Header	Cookies with access_token value
Body	{"state":6, "sysid":"<snow sys id>"}

Response (Success Case: 200 OK status code)	{
	    "status_code": 200,
	    "data":"",
	    "msg": ""
	}

Response (Failed Case 1: 403 Unauthorized status code)	{
	    "message": "Access denied"
	}

Response (Failed Case 2: 403 Unauthorized status code)	{
	    "message": "Invalid credentials"
	}

Response (Failed Case 3: 401 Unauthorized status code)	{
	    "detail":"Given token not valid for any token type",
	     "code":"token_not_valid",
	    "message": [
	{
	      "token_class":"AcessToken",
	      "token_type":"access",
	      "message":"Token is invalid or expired"
	}
	]

	}

Response
(Failed Case 4: 401 Unauthorized status code)
	{
	"detail":"Given token not valid for any token type",
	"code":"token_not_valid",
	"message": [
	{
	"token_class":"AcessToken",
	"token_type":"access",
	"message":"Token is invalid or expired"
	}
	]
	}

Response
(Failed Case 5: 401 Unauthorized status code)
	{
	"message": "Data has been sent for inactive customer"
	}

Response
(Failed Case 6: 401 Unauthorized status code)
	{
	"message": "User is not associated with the customer"
	}

4.4.3 API: SX Fetch Ticket Update

Table 17 – SX Fetch Ticket Update

Element	Description
API	Fetch Ticket Update
Description	This API is to update the actionable state in IEM whenever the ticket is getting resolved using the ITSM tool
Method	POST
URL	https://<Hosted_API>/<customerid>/update-ticket?integrationid=<integrationid>
Header	Cookies with access_token value
Body	{"state":1,"itemNumber":"<incident_number>","situationId":"<situation_id>"}

Response
(Success Case: 200 OK status code)
	{
	"status_code": 200,
	"data":"",
	"msg": ""
	}{

	"message": "Success" }

Response
(Failed Case 1: 403 Unauthorized status code)
	{
	"message": "Access denied"
	}

Response
(Failed Case 2: 403 Unauthorized status code)
	{
	"message": "Invalid credentials"
	}

Response
(Failed Case 3: 401 Unauthorized status code)
	{
	"detail":"Given token not valid for any token type",
	"code":"token_not_valid",
	"message": [
	{
	"token_class":"AcessToken",
	"token_type":"access",
	"message":"Token is invalid or expired"
	}
	]"Invalid access token"
	}

Response
(Failed Case 4: 401 Unauthorized status code)
	{
	"detail":"Given token not valid for any token type",
	"code":"token_not_valid",
	"message": [{
	"token_class":"AcessToken",
	"token_type":"access",
	"message":"Token is invalid or expired"
	}
	]
	}{
	"message": "Access token has expired"
	}

Response
(Failed Case 5: 401 Unauthorized status code)
	{
	"message": "Data has been sent for inactive customer"
	}

Response (Failed Case 6: 401 Unauthorized status code)	{ "message": "User is not associated with the customer"

	}

We need to replace <integrationid> with dynamic integration id which is generated by using MD5 hashing algorithm.

5 Data View API

To consume the functionality of IEM Dataview provided by a specific API, an enterprise tool needs to place a request. This section details the construction of that request that is supported by IEM Service. Requests are typically categorized in terms of the requested operation: create (POST) or retrieve (GET).

5.1 IEM Dataview API Details

5.1.1 Generate Token

Table 18 – Generate Token

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>API</td><td>Generate Token</td></tr>
<tr><td>Description</td><td>API returns the access token based on username and password to authenticate IEM Data view APIs.</td></tr>
<tr><td>Method</td><td>POST</td></tr>
<tr><td>URL</td><td>https://<Hosted_API>/api/v1/user/token</td></tr>
<tr><td>Body</td><td>NA</td></tr>
<tr><td>Header</td><td>Authorization headers using basic authentication with username and password</td></tr>
<tr><td>Response
(Success Case: 200 OK status code)</td><td><pre>
{
    "status_code": 200,
    "data": {
        "email": "<user_email_id>",
        "name": "<user_name>",
        "is_superadmin": false,
        "access_token": "<access_token_value>",
        "refresh_token": "<refresh_token_value>",
        "associated_with_customers": [
            {}
                        ]
    },
    "msg": ""
}
</pre></td></tr>
<tr><td>Response Parameter</td><td>Access_token</td></tr>
<tr><td>Response (Failed Case: 401 Unauthorized status code)</td><td><pre>
{
    "detail": "Invalid username/password."
}
</pre></td></tr>
</table>

5.1.2 Open actionable API

Table 19 – Open actionable Api

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>API</td><td>Open actionable API</td></tr>
<tr><td>Description</td><td>This API is used to fetch data related actionable based on various parameters and filters.</td></tr>
<tr><td>Method</td><td>GET</td></tr>
<tr><td>URL</td><td>https://<Hosted_API>/api/v1/<customerid>/data-item-data/v1?page=1&length=10&search=[]&user_selected_view=v1&filters=[{"p16,state.text,State NOT IN(Resolved,Closed)":{"field1":"p16,state.text,State","field2":"NOT IN","field3":"Resolved,Closed","field4":"","field5":"","field6":["Invalid Date","Invalid Date"]}}&usertimezone=Asia/Calcutta</td></tr>
<tr><td>Body</td><td>NA</td></tr>
<tr><td>Request URL Parameters</td><td>
<table>
<tr><td>Page</td><td>The page parameter is used to specify the page number for paginated results in the API request. If there is more data, it is divided into pages, with each page containing a subset of the data based on the specified length.</td></tr>
<tr><td>Length</td><td>The length parameter is used to specify the number of items per page.</td></tr>
<tr><td>Search</td><td>search which is used to filter results based on a search query.</td></tr>
<tr><td>user_selected_view</td><td>The user_selected_view parameter is used to specify the type of view the user wants to see, such as Actionable, Alerts, or Events</td></tr>
<tr><td>filter</td><td>The filter parameter is used to apply additional filtering criteria to the results by default we are filtering based on Resolved, Closed.</td></tr>
<tr><td>usertimezone</td><td>The usertimezone parameter is used to specify the user's timezone for accurate date and time representation.</td></tr>
</table>
</td></tr>
</table>

5.1.3 Open Actionable by First Occurrence API

Table 20 – Open Actionable by Filter Occurrence API

Element	Description
API	Open actionable by filter occurrence API
Description	This API is used to fetch data related actionable based on various parameters and filter as first occurrence.
Method	GET
URL	https://<Hosted_API>/api/v1/<customerid>/data-item-data/v1?page=1&length=10&search=[]&user_selected_view=v1&filters=[{"p16,state.te

<table>
<tr><td></td><td>xt,State NOT IN(Resolved,Closed)":{"field1":"p16,state.text,State","field2":"NOT IN","field3":"Resolved,Closed","field4":"","field5":"","field6":["Invalid Date","Invalid Date"]}},{"p88,$$tzstart$$firstoccurrence$$tzend$$,Pending Since IS(2025/01/15 11:57:00)":{"field1":"p88,$$tzstart$$firstoccurrence$$tzend$$,Pending Since","field2":"$$OPERATOR$$","field3":"","field4":"$$DATE$$","field5":"","field6":["Invalid Date","Invalid Date"]}}]&usertimezone=Asia/Calcutta</td></tr>
<tr><td>Body</td><td>NA</td></tr>
<tr><td>Request URL Parameters</td><td>
<table>
<tr><td>Page</td><td>The page parameter is used to specify the page number for paginated results in the API request. If there is more data, it is divided into pages, with each page containing a subset of the data based on the specified length.</td></tr>
<tr><td>Length</td><td>The length parameter is used to specify the number of items per page.</td></tr>
<tr><td>Search</td><td>search which is used to filter results based on a search query.</td></tr>
<tr><td>user_selected_view</td><td>The user_selected_view parameter is used to specify the type of view the user wants to see, such as Actionable, Alerts, or Events</td></tr>
<tr><td>filter</td><td>The filter parameter is used to apply additional filtering criteria to the results. Here we are filtering based on firstoccurrence.</td></tr>
<tr><td>usertimezone</td><td>The usertimezone parameter is used to specify the user's timezone for accurate date and time representation.</td></tr>
</table>
</td></tr>
</table>

5.1.4 Open Actionable by First Occurrence Filtering Between Data APIs

Table 21 – Open Actionable by First Occurrence Filtering Between Data APIs

Element	Description
API	Open Actionable by First Occurrence Filtering Between Data APIs
Description	This API is used to fetch data related actionable based on various parameters and filter with first occurrence data between time periods.
Method	GET
URL	https://<Hosted_API>/api/v1/<customerid>/data-item-data/v1?page=1&length=10&search=[]&user_selected_view=v1&filters=[{"p16,state.text,State NOT IN(Resolved,Closed)":{"field1":"p16,state.text,State","field2":"NOT IN","field3":"Resolved,Closed","field4":"","field5":"","field6":["Invalid Date","Invalid Date"]}},{"p88,$$tzstart$$firstoccurrence$$tzend$$,Pending Since +between(2025/01/15 12:15:00 and 2025/01/15 17:15:00)":{"field1":"p88,$$tzstart$$firstoccurrence$$tzend$$,Pending Since","field2":"$$OPERATOR$$","field3":"","field4":"$$FROM_DATE$$","field5":"$$TO_DATE$$"","field6":["Invalid Date","Invalid Date"]}}]&usertimezone=Asia/Calcutta

Body	NA	
Request URL Parameters	Page	The page parameter is used to specify the page number for paginated results in the API request. If there is more data, it is divided into pages, with each page containing a subset of the data based on the specified length.
	Length	The length parameter is used to specify the number of items per page.
	Search	search which is used to filter results based on a search query.
	user_selected_view	The user_selected_view parameter is used to specify the type of view the user wants to see, such as Actionable, Alerts, or Events
	filter	The filter parameter is used to apply additional filtering criteria to the results. Here we are filtering based on firstoccurrence with between time periods.
	usertimezone	The usertimezone parameter is used to specify the user's timezone for accurate date and time representation.

5.1.5 Open Actionable by Last Occurrence API

Table 22 – Open Actionable by Last Occurrence API

Element	Description	
API	Open actionable by last occurrence API	
Description	This API is used to fetch data related actionable based on various parameters and filter with last occurrence data.	
Method	GET	
URL	https://<Hosted_API>/api/v1/<customerid>/data-item-data/v1?page=1&length=10&search=[]&user_selected_view=v1&filters=[{"p16,state.text,State+NOT IN(Resolved,Closed)":{"field1":"p16,state.text,State","field2":"NOT IN","field3":"Resolved,Closed","field4":"","field5":"","field6":["Invalid+Date","Invalid+Date"]}},{"p88,$$tzstart$$lastoccurrence$$tzend$$,Last Updated+IS(2023/12/18 14:29:43)":{"field1":"p88,$$tzstart$$lastoccurrence$$tzend$$,Last Updated","field2":"$$OPERATOR$$","field3":"","field4":"$$DATE$$","field5":"","field6":["Invalid+Date","Invalid+Date"]}}]&usertimezone=Asia/Calcutta	
Body	NA	
Request URL Parameters	Page	The page parameter is used to specify the page number for paginated results in the API request. If there is more data, it is divided into pages, with each page containing a subset of the data based on the specified length.
	Length	The length parameter is used to specify the number of items per page.
	Search	search which is used to filter results based on a search query.

<table>
<tr><td rowspan="3"></td><td>user_selected_view</td><td>The user_selected_view parameter is used to specify the type of view the user wants to see, such as Actionable, Alerts, or Events</td></tr>
<tr><td>filter</td><td>The filter parameter is used to apply additional filtering criteria to the results. Here we are filtering based on last occurrence.</td></tr>
<tr><td>usertimezone</td><td>The usertimezone parameter is used to specify the user's timezone for accurate date and time representation.</td></tr>
</table>

5.1.6 Get Related Alerts API

Table 23 –Get Related Alerts API

<table>
<tr><th>Element</th><th colspan="2">Description</th></tr>
<tr><td>API</td><td colspan="2">Get related alerts API</td></tr>
<tr><td>Description</td><td colspan="2">This API is used to fetch alerts data which is related to the particular actionables based on various parameters.</td></tr>
<tr><td>Method</td><td colspan="2">GET</td></tr>
<tr><td>URL</td><td colspan="2">https://<Hosted_API>/api/v1/<customerid>/data-item-data/v1/<actionableid>/related-data?page=1&length=10&search=[]&&usertimezone=Asia%2FCalcutta</td></tr>
<tr><td>Body</td><td colspan="2">NA</td></tr>
<tr><td rowspan="4">Request URL Parameters</td><td>Page</td><td>The page parameter is used to specify the page number for paginated results in the API request. If there is more data, it is divided into pages, with each page containing a subset of the data based on the specified length.</td></tr>
<tr><td>Length</td><td>The length parameter is used to specify the number of items per page.</td></tr>
<tr><td>Search</td><td>search which is used to filter results based on a search query.</td></tr>
<tr><td>usertimezone</td><td>The usertimezone parameter is used to specify the user's timezone for accurate date and time representation.</td></tr>
</table>

5.1.7 Get Related Events API

Table 24 – Get Related Events API

<table>
<tr><th>Element</th><th colspan="2">Description</th></tr>
<tr><td>API</td><td colspan="2">Get related events API</td></tr>
<tr><td>Description</td><td colspan="2">This API is used to fetch events data which is related to the Alert based on various parameters.</td></tr>
<tr><td>Method</td><td colspan="2">GET</td></tr>
<tr><td>URL</td><td colspan="2">https://<Hosted_API>/api/v1/<customerid>/data-item-data/v2/<alertid>/related-data?page=1&length=10&usertimezone=Asia/Calcutta</td></tr>
<tr><td>Body</td><td colspan="2">NA</td></tr>
<tr><td>Request URL Parameters</td><td>Page</td><td>The page parameter is used to specify the page number for paginated results in the API request. If there is more data, it is divided into pages, with each page containing a subset of the data based on the specified length.</td></tr>
</table>

Length	The length parameter is used to specify the number of items per page.
Search	search which is used to filter results based on a search query.
usertimezone	The usertimezone parameter is used to specify the user's timezone for accurate date and time representation.

5.1.8 Open Actionable by User API

Table 25 – Open Actionable by User API

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>API</td><td>Open actionable by user API</td></tr>
<tr><td>Description</td><td>This API is used to fetch the actionable data which is assigned to the user based on various parameters.</td></tr>
<tr><td>Method</td><td>GET</td></tr>
<tr><td>URL</td><td>https://<Hosted_API>/api/v1/<customerid>/data-item-data/v1?page=1&length=10&search=[]&filters=[{"p16,state.text,State+NOT IN(Resolved,Closed)":{"field1":"p16,state.text,State","field2":"NOT IN","field3":"Resolved,Closed","field4":"","field5":"","field6":["Invalid+Date","Invalid+Date"]}},{"p16,assignedusers,AssignedUsers":{"field1":"p16,assignedusers,AssignedUser","field2":"IN","field3":"$$ASSIGNED_USERS$$","field4":"","field5":"","field6":["Invalid+Date","Invalid+Date"]}}]&user_selected_view=v1&usertimezone=Asia/Calcutta</td></tr>
<tr><td>Body</td><td>NA</td></tr>
<tr><td>Request URL Parameters</td><td><table>
<tr><td>Page</td><td>The page parameter is used to specify the page number for paginated results in the API request. If there is more data, it is divided into pages, with each page containing a subset of the data based on the specified length.</td></tr>
<tr><td>Length</td><td>The length parameter is used to specify the number of items per page.</td></tr>
<tr><td>Search</td><td>search which is used to filter results based on a search query.</td></tr>
<tr><td>user_selected_view</td><td>The user_selected_view parameter is used to specify the type of view the user wants to see, such as Actionable, Alerts, or Events</td></tr>
<tr><td>filter</td><td>The filter parameter is used to apply additional filtering criteria to the results. Here we are filtering based on AssignedUser.</td></tr>
<tr><td>usertimezone</td><td>The usertimezone parameter is used to specify the user's timezone for accurate date and time representation.</td></tr>
</table></td></tr>
</table>

5.1.9 Entity Data API

Table 26 – Entity Data API

Element	Description

<table>
<tr><td>API</td><td>Entity Data API</td></tr>
<tr><td>Description</td><td>This API is used to fetch the entity data based on various parameters.</td></tr>
<tr><td>Method</td><td>GET</td></tr>
<tr><td>URL</td><td>https://<Hosted_API>/api/v1/<customerid>/entity?page=1&length=10&usertimezone=Asia/Calcutta</td></tr>
<tr><td>Body</td><td>NA</td></tr>
<tr><td>Request URL Parameters</td><td><table>
<tr><td>Page</td><td>The page parameter is used to specify the page number for paginated results in the API request. If there is more data, it is divided into pages, with each page containing a subset of the data based on the specified length.</td></tr>
<tr><td>Length</td><td>The length parameter is used to specify the number of items per page.</td></tr>
<tr><td>usertimezone</td><td>The usertimezone parameter is used to specify the user's timezone for accurate date and time representation.</td></tr>
</table></td></tr>
</table>

5.1.10 API: AEX integration

Table 27 – AEX integration Api

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>API</td><td>AEX integration API</td></tr>
<tr><td>Description</td><td>This API is to update the actionable state in IEM whenever the ticket is getting resolved using the ITSM tool.</td></tr>
<tr><td>Method</td><td>GET</td></tr>
<tr><td>URL</td><td>https://<Hosted_API>/api/v1/<customerid>/data-item-data/v1?page=1&length=10&search=[][]&&column_text=argMax(vactionable.severity_desc,vactionable.inserttimestamp) as Severity, vactionable.iactionableid as ID, argMax(vactionable.state_desc,vactionable.inserttimestamp) as State, argMax(vactionable.entity,vactionable.inserttimestamp) as Entity, argMax(vactionable.description,vactionable.inserttimestamp) as Description, $$usercustom$$snowsysid^ as External Ticket Number,&filters=[{"p88,vactionable.iactionableid,ID IS({{actionable_id}})":{"field1":"vactionable.iactionableid","field2":"IS","field3":"","field4":{{actionable_id}}}}]&user_selected_view=v1&usertimezone=Asia/Calcutta</td></tr>
<tr><td>Body</td><td>NA</td></tr>
<tr><td>Request URL Parameters</td><td><table>
<tr><td>Page</td><td>The page parameter is used to specify the page number for paginated results in the API request. If there is more data, it is</td></tr>
</table></td></tr>
</table>

	divided into pages, with each page containing a subset of the data based on the specified length.
Length	The length parameter is used to specify the number of items per page.
Search	search which is used to filter results based on a search query.
user_selected_view	The user_selected_view parameter is used to specify the type of view the user wants to see, such as Actionable, Alerts, or Events
usertimezone	The usertimezone parameter is used to specify the user's timezone for accurate date and time representation.

1. In the dataview api, we have used filter values like this $$OPERATOR$$, while calling the apis to replace this place holder with operator values like ">","<","between",">=","<=".

2. If the operator value is BETWEEN it has two variables, one is $$FROM_DATE$$ and $$TO_DATE$$,

the remaining operator will have only $$DATE$$.

3. For $$ASSIGNED_USERS$$ field, we can replace with email id like ["user1@hcl.com"]

4. For <customerid> field, we must replace with envcustomerid which will be generated from token API [https://<Hosted_API>/api/v1/user/token]

5. For <actionableid> field, we have to replace it with an ID which will be obtained from listing API [https://<Hosted_API>/api/v1/<customerid>/data-item-data/v1?page=1&length=10&search=[]&user_selected_view=v1&filters=[]&usertimezone=Asia/Calcutta].

6. For <alertid> field, we have to replace it with an ID which will be obtained from the listing API [https://<Hosted_API>/api/v1/<customerid>/data-item-data/v2?page=1&length=10&search=[]&user_selected_view=v1&filters=[]&usertimezone=Asia/Calcutta].

6 Noise Maintenance Api

6.1 Login Details

Noise Maintenance API supports token authentication as follows. For further details 4.3.1API Authentication.

6.1.1 Generate Token

Table 28 – Generate Token

Element	Description
API	Generate Token
Description	API returns the access token based on username and password to authenticate IEM Data view APIs.
Method	POST
URL	https://<Hosted_API>/api/v1/user/token
Body	NA
Header	Authorization headers using basic authentication with username and password

Response (Success Case: 200 OK status code)	{
	    "status_code": 200,
	    "data": {
	        "email": "<user_email_id>",
	        "name": "<user_name>",
	        "is_superadmin": false,
	        "access_token": "<access_token_value>",
	        "refresh_token": "<refresh_token_value>",
	        "associated_with_customers": [
	            {}
	                        ]
	    },
	    "msg": ""
	}

Response Parameter	Access_token

Response (Failed Case: 401 Unauthorized status code)	{
	    "detail": "Invalid username/password."
	}

6.2 Calling Noise Maintenance APIs

6.2.1 Base Filter with Define Maintenance and Without recurring

Table 29 – Noise Rule creation

Element	Description
API	Noise Rule Creation
Description	This Api is to create the noise with define maintenance and without recurring
Method	POST
URL	https://<Hosted_API>/<customerid>/noise

Body	{
	"rule_name": "<noise_name>",
	"param_fields": [<base filter>],
	"custom_param_fields": [],
	"start_date": "<from_date>",
	"end_date": "<to_date>",
	"start_time": "<start_time>",
	"end_time": "<end_time>",
	"is_recurring": "N",
	"user_timezone_save_info": {},
	"repeatevery": "",
	"repeatonvalue": "",
	"define_maintenance": "Y"
	}

Header	Cookies with access_token value

Response (Success Case: 200 OK status code)	{
	"status_code": 200,
	"data": "",
	"msg": "noise rule created successfully"
	}

Response (Failed Case 1: 401 Unauthorized status code)	{
	"detail":"Given token not valid for any token type",
	"code":"token_not_valid",
	"message": [
	{
	"token_class":"AcessToken",
	"token_type":"access",
	"message":"Token is invalid or expired"
	}

	] }

Response (Failed Case 2: 400 Unauthorized status code)	{
	"status_code":400,
	"data": "",
	"msg": "noise rule already exists"
	}

6.2.2 Base Filter and Custom Filter with Define Maintenance and Without recurring

Table 30 – Noise Rule creation

Element	Description
API	Noise Rule Creation
Description	This Api is to create the noise with base and custom filters with define maintenance
Method	POST
URL	https://<Hosted_API>/<customerid>/noise

Body	{
	"rule_name": "<noise_name>",
	"param_fields": [<base filter>],
	"custom_param_fields": [<custom filter>],
	"start_date": "<from_date>",
	"end_date": "<to_date>",
	"start_time": "<start_time>",
	"end_time": "<end_time>",
	"is_recurring": "N",
	"user_timezone_save_info": {},
	"repeatevery": "",
	"repeatonvalue": "",
	"define_maintenance": "Y"
	}

Header	Cookies with access_token value

Response (Success Case: 200 OK status code)	{
	"status_code": 200,
	"data": "",
	"msg": "noise rule created successfully"
	}

Response (Failed Case 1: 401	{

Unauthorized status code)	"detail":"Given token not valid for any token type",
	"code":"token_not_valid",
	"message": [
	{
	"token_class":"AcessToken",
	"token_type":"access",
	"message":"Token is invalid or expired"
	}
	]
	}

Response (Failed Case 2: 400 Unauthorized status code)	{
	"status_code":400,
	"data": "",
	"msg": "noise rule already exists"
	}

6.2.3 Base Filter with Define Maintenance and with Recurring

1. Recurring Daily

Table 31 – Noise Rule creation

Element	Description
API	Noise Rule Creation
Description	This Api is to create the noise with base and custom filters with recurring on daily.
Method	POST
URL	https://<Hosted_API>/<CUSTOMERID>/noise

Body	{
	"rule_name": "<noise_name>",
	"param_fields": [<base filter>],
	"custom_param_fields": [<custom filter>],
	"start_date": "<from_date>",
	"end_date": "<to_date>",
	"start_time": "<start_time>",
	"end_time": "<end_time>",
	"is_recurring": "Y ",
	"user_timezone_save_info": {
	"start_date": "<from_date>",
	"end_date": "<to_date>",
	"start_time": "<start_time>",

	"end_time": "<end_time>", }, "repeatevery": "daily", "repeatonvalue": "", "define_maintenance": "Y" }
Header	Cookies with access_token value

Response (Success Case: 200 OK status code)	{
	"status_code": 200,
	"data": "",
	"msg": "noise rule created successfully"
	}

Response (Failed Case 1: 401 Unauthorized status code)	{
	"detail":"Given token not valid for any token type",
	"code":"token_not_valid",
	"message": [
	{
	"token_class":"AcessToken",
	"token_type":"access",
	"message":"Token is invalid or expired"
	}
	]
	}

Response (Failed Case 2: 400 Unauthorized status code)	{
	"status_code":400,
	"data": "",
	"msg": "noise rule already exists"
	}

2. Recurring Weekly

Table 32 – Noise Rule creation

Element	Description
API	Noise Rule Creation
Description	This Api is to create the noise with base and custom filters with define maintenance and without recurring on weekly
Method	POST
URL	https://<Hosted_API>/<customerid>/noise
Body	{ "rule_name": "<noise_name>",

	"param_fields": [<base filter>], "custom_param_fields": [<custom filter>], "start_date": "<from_date>", "end_date": "<to_date>", "start_time": "<start_time>", "end_time": "<end_time>", "is_recurring": "Y", "user_timezone_save_info": {}, "repeatevery": "weekly", "repeatonvalue": [<recursing_day>], "define_maintenance": "Y" }
Header	Cookies with access_token value

Response (Success Case: 200 OK status code)	{
	"status_code": 200,
	"data": "",
	"msg": "noise rule created successfully"
	}

Response (Failed Case 1: 401 Unauthorized status code)	{
	"detail":"Given token not valid for any token type",
	"code":"token_not_valid",
	"message": [
	{
	"token_class":"AcessToken",
	"token_type":"access",
	"message":"Token is invalid or expired"
	}
	]
	}

Response (Failed Case 2: 400 Unauthorized status code)	{
	"status_code":400,
	"data": "",
	"msg": "noise rule already exists"
	}

3. Recurring Monthly

Table 33 – Noise Rule creation

Element	Description
API	Noise Rule Creation

Description	This Api is to create the noise with base and custom filters with define maintenance and without recurring on monthly
Method	POST
URL	https://<Hosted_API>/<customerid>/noise

Body	{
	    "rule_name": "<noise_name>",
	    "param_fields": [<base filter>],
	    "custom_param_fields": [<custom filter>],
	    "start_date": "<from_date>",
	    "end_date": "<to_date>",
	    "start_time": "<start_time>",
	    "end_time": "<end_time>",
	    "is_recurring": "Y",
	    "user_timezone_save_info": {
	      "start_date": "<from_date>",
	      "end_date": "<to_date>",
	      "start_time": "<start_time>",
	      "end_time": "<end_time>",
	      },
	    "repeatevery": "monthly",
	    "repeatonvalue": [<recursing_day>],
	    "define_maintenance": "Y"
	}

Header	Cookies with access_token value

Response (Success Case: 200 OK status code)	{
	    "status_code": 200,
	    "data": "",
	    "msg": "noise rule created successfully"
	}

Response (Failed Case 1: 401 Unauthorized status code)	{
	    "detail":"Given token not valid for any token type",
	     "code":"token_not_valid",
	    "message": [
	{
	      "token_class":"AcessToken",
	      "token_type":"access",
	      "message":"Token is invalid or expired"
	   }
	]

		}

Response (Failed Case 2: 400 Unauthorized status code)		{
		    "status_code":400,
		    "data": "",
		    "msg": "noise rule already exists"
		}

In the noise maintenance api, we must put the noise name in the <noise_name> place holder.

For <base filter>, we must include base filter object data which consists of a mandatory filter like [{"parameter": "agentlocation","operator": "LIKE","value": "agent"}].

For <custom filter>, we must include customized filter object data which consists of customized filters like [{"parameter": "country","operator": "LIKE","value": "india"}].

<from_date> and <to_date> must be replaced with starting and ending dates [like "2025-01-20"

].

<start_time> and <end_time> must be replaced with starting and ending timings [like "00:00"

].

<recursing_day> used to give days in weekly recursion like ["1", "2”]

<recursing_date> needs to replace the date between starting and ending date which was given between <from_date> and <to_date>.

7 API Accessible Matrix

Table 34 - API Accessible Matrix

Type	Action	API User	Organization Users
Provisioning Request	Get	User should be part of organization, but can't login into UI of IEM User can only perform the set of API actions	User should be part of organization and user can login into UI of IEM but cannot access these APIs
	Post	User should be part of organization, but can't login into UI of IEM User can only perform the set of API actions	User should be part of organization and user can login into UI of IEM but cannot access these APIs

HCLSoftware

hcltechsw.com