axway

Sentinel

Version 4.2.0

01 October 2025

Open API Developer Guide

This documentation describes the following Axway software:

Axway Sentinel 4.2.0

Contents

Accessibility

Axway strives to create accessible products and documentation for users.

The product documentation provides the following accessibility features:

- Screen reader support
- Support for high contrast and accessible use of colors

Screen reader support

- Alternative text is provided for images whenever necessary.
- The PDF documents are tagged to provide a logical reading order.

Support for high contrast and accessible use of colors

- The documentation can be used in high-contrast mode.
- There is sufficient contrast between the text and the background color.

Preface

Axway Sentinel is introducing the innovative strategic initiative "Open Sentinel" to bolster support for Axway's Sentinel user base in maximizing the value and insights derived from QLT messages generated by Axway's MFT technologies. This approach empowers customers to autonomously access and derive analytics and insights from evolving QLT messages using established third-party monitoring and analytics tools. Open Sentinel streamlines the retrieval of QLT messages stored in Sentinel's database. This initiative marks a significant step forward in providing users with greater flexibility and control over their data analysis and decision-making processes within the Axway Sentinel ecosystem.

About Sentinel Open API

This guide describes the Axway Sentinel "Open API" presents an innovative alternative to the established Public API, which is currently dependent on Java RMI technology. However, it is imperative to bridge this divergence in order to facilitate Sentinel's utilization of pre-defined queries within the Sentinel environment for integration with external third-party tools. The implementation of the Sentinel Open API for the Sentinel Web Dashboard will involve the exposure of this interface from the Web Dashboard component within the Jetty server infrastructure. This advancement will effectively facilitate the seamless integration of Sentinel with a diverse range of existing third-party tools, thereby empowering users with enhanced flexibility and expanded functionality.

Authenticate HTTP requests

The Sentinel JSON API requires basic authentication. This means HTTP requests must be authenticated with a user and password corresponding to a valid Sentinel user.

In addition to the "Authorization" header parameter used for basic authentication, each HTTP request must specify the header parameter "Accept" with the value `application/json`. Depending on the type or expected response, Sentinel responds in JSON format.

Some requests will require body parameters. These can also be supplied in JSON formats. Depending on the chosen format, whenever body parameters are required, an additional Header parameter is necessary indicating the format in which the Body parameters are supplied. You must set the Header parameter "**Content-Type**" with the value **`application/json`**; this is also dependent upon your choice of the format. When user use any third party tool other than swagger UI then you must set the Header parameter "**Referer**" with the value `<hostname:1309/Sentinelwebdashboard/`

Note The examples in this document assume that Sentinel is installed on the host `acme.com` with the default ports.

Who should read this guide

This guide is intended for developers who are responsible for Sentinel API development.

Other technical or business users may find parts of this guide useful and informative as well.

How to use this guide

The information in this guide is designed to be used in conjunction with, but does not replace, the product documentation.

Before you begin, it is recommended to review this guide thoroughly. The following is a brief description of the contents of each chapter:

Sentinel Open API Requests – Describes the services used to list and execute Requests deployed on Sentinel.

Known limitations – Describes the known limitations for the Sentinel API.

Sentinel documentation set

The Sentinel documentation set includes the following documents:

- *Axway Sentinel Installation Guide*

  Explains how to install Sentinel using the Axway Installer and how to upgrade Sentinel to the latest version.

- *Axway Sentinel Configuration Guide*

  Describes how to configure and start up Sentinel.

- Axway Sentinel *Monitoring User Guide*

  Describes how to use and monitor Sentinel.

- Axway Sentinel *Web Dashboard User Guide*

  Describes how to create and use Sentinel Web Dashboards.

- *Axway Sentinel API Developer Guide*

  Describes how to use the Sentinel REST API as an alternative to the already existing Public API.

  Axway Sentinel Open API Developer Guide

  Describes how to use the Sentinel Open API as an alternative to the already existing REST/Public API.

- *Axway Sentinel Release Notes*

  Provides up-to-date information related to new features included in this release and known limitations.

- *Axway Sentinel Security Guide*

  Provides instructions and recommendations to help you strengthen the security of Axway Sentinel and of the global solution.

More Axway documentation is available on the Axway Documentation portal at docs.axway.com.

Training services

Axway offers training across the globe, including on-site instructor-led classes and self-paced online learning. For details, go to Axway University at university.axway.com/learn

Support services

Go to Axway Support at support.axway.com for:

- Product downloads, service packs and patches
- Knowledge base articles
- Access to your cases

The Axway Global Support team provides worldwide 24 x 7 support, subject to validation of your license agreement. Email support@axway.com.

Sentinel Open API requests 1

This section provides a framework for documenting the OpenAPI based services that operate across Sentinel different modules. Its purpose is to ensure that every module which offers API functionality has a clear, uniform way of exposing, describing, and using its endpoints. Whether for internal integrations or external consumers, consistency in API schemas, request/response formats, and type definitions helps prevent confusion and speeds up development.

What this Section Covers

• **Available Services**: What OpenAPI endpoints are exposed in each module (create, read, update, delete, reports, etc.).

• **Request & Response Formats:** How requests must be structured (JSON body, query parameters, path parameters) and what responses look like (status codes, content types, possible error messages).

• **Parameter Types:** Definition of parameter fields, attributes, and allowed types (for example, String, Boolean, Integer, etc.), including when and how to use user variable parameters.

• **Module Relevance:** Which endpoints apply to which modules — not every module supports every endpoint. The documentation will clarify module applicability.

• **Examples & Sample Payloads:** Realistic sample requests/responses to guide developers, helping avoid trial and error.

The following OpenAPI-based services are available across Sentinel different modules:

- *getReportNameList* on page 10
- *getReportDataByName* on page 13
- *getGroupDetails* on page 25
- *getUserDetails* on page 23
- *CreateUser* on page 26

Rights and privileges

The user specified for each invocation of the API must be authorized to view and execute Sentinel Requests. Depending on the employed access manager, the following rights/privileges are required to access Sentinel Requests using the API:

- **Sentinel Administration**: Access to the information using Rest API to Execute Requests and access new Administration UI and Webdashboard.

- **Sentinel WebDashboard**: Access to the information, Deploy and remove objects using Rest API to Execute Requests and access of Webdashboard.

getReportNameList

This method displays the list of all existing Reports.

Request type and path

`GET: /sentinel/api/v1/report`

Parameters

This method does not use additional parameters besides the common parameters for this section.

sorted_by:string

You have the option to sort the data by NAME_ASC, NAME_DESC, MODIFICATION_DATE_ASC, MODIFICATION_DATE_DESC, CREATION_DATE_ASC, or CREATION_DATE_DESC. By default, the sorting is set to MODIFICATION_DATE_DESC

JSON request sample

1) By default , When the user doesn't inputs any available value corresponding to "**sorted_by:string**", the default sorting is set to MODIFICATION_DATE_DESC .

Request

<Sentinel_home>/SentinelWebDashboard/sentinel/api/v1/report

Responses

The following are the responses for this request path: **`GET:`** `/sentinel/api/v1/report`

Response


```
{
"header": null,
"data": [
{
"type": "List Of Reports",
"id": "101",
"attributes": {
"modificationDate": "2024-01-15 06:49:04.0",
"createdDate": "2023-12-14 11:38:23.0",
"reportName": "report_sql_svggen",
```

```
"queryOID": "102",
"ReportType": "0"
},
"relationships": null,
"links": null
},
{
"type": "List Of Reports",
"id": "100",
"attributes": {
"modificationDate": "2024-01-15 06:48:38.0",
"createdDate": "2023-12-14 11:24:59.0",
"reportName": "report_sdd_svggen",
"queryOID": "100",
"ReportType": "0"
},
"relationships": null,
"links": null
},
{
"type": "List Of Reports",
"id": "102",
"attributes": {
"modificationDate": "2024-01-05 12:03:12.0",
"createdDate": "2024-01-03 10:10:03.0",
"reportName": "demo_sql_report1",
"queryOID": "104",
"ReportType": "0"
},
"relationships": null,
"links": null
}
],
"included": null,
"links": null,
"meta": null
}
```


2) When the user inputs the available value corresponding to "**sorted_by:string**" .

Request

<sentinel_home>/SentinelWebDashboard/sentinel/api/v1/report?sorted_by=MODIFICATION_DATE_DESC

Responses

The following are the responses for this request path: **GET:** `/sentinel/api/v1/report`

Response


```
{
"header": null,
"data": [
{
"type": "List Of Reports",
"id": "101",
"attributes": {
"modificationDate": "2024-01-15 06:49:04.0",
"createdDate": "2023-12-14 11:38:23.0",
"reportName": "report_sql_svggen",
"queryOID": "102",
"ReportType": "0"
},
"relationships": null,
"links": null
},
{
"type": "List Of Reports",
"id": "100",
"attributes": {
"modificationDate": "2024-01-15 06:48:38.0",
"createdDate": "2023-12-14 11:24:59.0",
"reportName": "report_sdd_svggen",
"queryOID": "100",
"ReportType": "0"
},
"relationships": null,
"links": null
},
{
"type": "List Of Reports",
"id": "102",
"attributes": {
"modificationDate": "2024-01-05 12:03:12.0",
"createdDate": "2024-01-03 10:10:03.0",
"reportName": "demo_sql_report1",
"queryOID": "104",
"ReportType": "0"
},
"relationships": null,
"links": null
}
],
```

```
"included": null,
"links": null,
"meta": null
}
```

getReportDataByName

This method displays the report data for a provided Report Name.

Request type and path

`POST: /sentinel/api/v1/report/{reportName}`

Where `{reportName}` is the reportName of the Report. The reportName can be retrieved using the "getReportNameList" method.

Parameters

This method does not use additional parameters besides the common parameters for this section.

"**attributes**": null

Note: Please specify the desired column name in the 'attributes' parameter to display the result corresponding to the specified column name. Otherwise, set the attribute value to 'null' to display all columns selected by default. Additionally, the user can add multiple filters by using a comma separator like "**attributes**": "CycleId,EventDate", .

"**dir**": "string",

Note: This field will be implemented further in future.

"**fields**": {},

Note: This field will be implemented further in future.

"**include**": "string",

Note: This field will be implemented further in future.

"**page**": 1,

Note: By default "Page" value is '0' and display the selected page out of the total number of pages in pagination mode.

"**report_filter**":[

Note: The Report Filter only works with defined filters on the SQL data dictionary. In the absence of a defined filter, all data is displayed based on the SQL query.

{

"**columnName**": "string",

"**columnValue**": {}

}

Note: In the POST request, users have the option to define filter parameters inside the report filter tag, corresponding to the specified attributes such as "ColumnName" and "ColumnValue". These defined parameters have already been incorporated into the SQL query that corresponds to the Reportname mentioned in the POST request. Users can directly input these parameters values from the currently used database via corresponding tables. Additionally, users can add multiple filters by using a comma separator, and all of them will function according to the already defined logical operator between different parameters in the SQL Query.

]

"**size**": 10,

Note: By default "Size" value is '0' and Display the number of items or records per page

"**sort**": "string"

Note: This field will be implemented further in future.

JSON request sample

1) When user didn't change any value in POST, request body and execute the request with reportName.

Note: When no valid parameters are passed that match those available parameters in the request, then you will not get proper data in the response.

Request

<Sentinel_Home>/SentinelWebDashboard/sentinel/api/v1/report/demo_sql_report1

Request Body

```
{
"attributes": "string",
"dir": "string",
"fields": {},
"include": "string",
"page": 0,
"report_filter": [
{
"columnName": "string",
"columnValue": {}
}
],
"size": 0,
"sort": "string"
}
```

Responses

The following are the responses for this request path: **`POST`**:
`/sentinel/api/v1/report/reportName`

where **`reportName`** is the id of a reportName like `TestReport`

Response

```
{
"header": {},
"data": [
{
"type": "Report Resource",
"id": "1",
"attributes": {},
"relationships": null,
"links": null
},
{
"type": "Report Resource",
"id": "2",
"attributes": {},
"relationships": null,
"links": null
},
.....
{
"type": "Report Resource",
"id": "1000",
"attributes": {},
"relationships": null,
"links": null
}
],
"included": null,
"links": null,
"meta": null
}
```

2) When user input null value corresponding attribute in POST, request body and execute the request with reportName.

Request

<Sentinel_Home>/SentinelWebDashboard/sentinel/api/v1/report/demo_sql_report1

Request Body

```
{
"attributes": null,
"dir": "string",
"fields": {},
"include": "string",
"page": 0,
"report_filter": [
{
"columnName": "string",
"columnValue": {}
}
],
"size": 0,
"sort": "string"
}
```

The following are the responses for this request path: **`POST`**:
`/sentinel/api/v1/report/reportName`

where **`reportName`** is the id of a reportName like `TestReport`

Response

```
{
"header": {},
"data": [
{
"type": "Report Resource",
"id": "1",
"attributes": {
"EVENTTIME": "16:12:11",
"ISALERT": "3",
"CYCLEID": "Test1",
"EVENTID": "101"
},
"relationships": null,
"links": null
},
{
"type": "Report Resource",
"id": "2",
"attributes": {
"EVENTTIME": "16:12:12",
"ISALERT": "1",
"CYCLEID": "Test2",
"EVENTID": "201"
},
"relationships": null,
```

```
"links": null
},
{
"type": "Report Resource",
"id": "3",
"attributes": {
"EVENTTIME": "16:12:13",
"ISALERT": "1",
"CYCLEID": "Test3",
"EVENTID": "301"
},
"relationships": null,
"links": null
},
.....
{
"type": "Report Resource",
"id": "1000",
"attributes": {
"EVENTTIME": "16:25:42",
"ISALERT": "1",
"CYCLEID": "Test1000",
"EVENTID": "100001"
},
"relationships": null,
"links": null
}
],
"included": null,
"links": null,
"meta": null
}
```


3) When user input value corresponding attributes attribute in POST, request body and execute the request with reportName.

Request

<Sentinel_Home>/SentinelWebDashboard/sentinel/api/v1/report/demo_sql_report1

Request Body


```
{
"attributes":"CycleId",
"dir": "string",
"fields": {},
"include": "string",
"page": 0,
```

```
"report_filter": [
{
"columnName": "string",
"columnValue": {}
}
],
"size": 0,
"sort": "string"
}
```

The following are the responses for this request path: **`POST`**:
`/sentinel/api/v1/report/reportName`

where **`reportName`** is the id of a reportName like `TestReport`

Response

```
{
"header": {},
"data": [
{
"type": "Report Resource",
"id": "1",
"attributes": {
"CYCLEID": "Test1"
},
"relationships": null,
"links": null
},
{
"type": "Report Resource",
"id": "2",
"attributes": {
"CYCLEID": "Test2"
},
"relationships": null,
"links": null
},
{
"type": "Report Resource",
"id": "3",
"attributes": {
"CYCLEID": "Test3"
},
"relationships": null,
"links": null
},
.....
```

```
{
"type": "Report Resource",
"id": "1000",
"attributes": {
"CYCLEID": "Test1000"
},
"relationships": null,
"links": null
}
],
"included": null,
"links": null,
"meta": null
}
```


Meanwhile if user input null value in attribute then it display default column present in Report.

4) When user input values corresponding to report filter in POST request body and execute the request with reportName

Request

<Sentinel_Home>/SentinelWebDashboard/sentinel/api/v1/report/demo_sql_report1

Request Body


```
{
"attributes": "CYCLEID,EVENTDATE",
"dir": "string",
"fields": {},
"include": "string",
"page": 1,
"report_filter": [
{
"columnName": "EventDate",
"columnValue": "18-09-23"
}
],
"size": 10,
"sort": "string"
}
```


The following are the responses for this request path: **`POST`**:
`/sentinel/api/v1/report/reportName`

where **`reportName`** is the id of a reportName like `TestReport`

Response


```
{
```

```
"header": {},
"data": [
{
"type": "Report Resource",
"id": "1",
"attributes": {
"CYCLEID": "Test1",
"EVENTDATE": "2023-09-18 00:00:00.0"
},
"relationships": null,
"links": null
},
{
"type": "Report Resource",
"id": "2",
"attributes": {
"CYCLEID": "Test2",
"EVENTDATE": "2023-09-18 00:00:00.0"
},
"relationships": null,
"links": null
},
{
"type": "Report Resource",
"id": "3",
"attributes": {
"CYCLEID": "Test3",
"EVENTDATE": "2023-09-18 00:00:00.0"
},
"relationships": null,
"links": null
},
.....
{
"type": "Report Resource",
"id": "1000",
"attributes": {
"CYCLEID": "Test1000",
"EVENTDATE": "2023-09-18 00:00:00.0"
},
"relationships": null,
"links": null
}
],
"included": null,
"links": {
"self": "WebDashboard/sentinel/api/v1/report/demo_sql_report1?page=1",
"next": "WebDashboard/sentinel/api/v1/report/demo_sql_report1?page=2",
```

```
"last": "WebDashboard/sentinel/api/v1/report/demo_sql_report1?page=100",
"first": null,
"prev": null
},
"meta": {
"count": 100,
"totalItems": null
}
}
```


5) When user input multiple values corresponding to report filter in POST request body and execute the request with reportName

Request

<Sentinel_Home>/SentinelWebDashboard/sentinel/api/v1/report/demo_sql_report1

Request Body


```
{
"attributes": "string",
"dir": "string", "fields": {},
"include": "string",
"page": 0,
"report_filter": [
{
"columnName": "EventDate1",
"columnValue": "5/4/2025"
},
{
"columnName": "EventDate2",
"columnValue": "5/14/2025"
}
],
"size": 0,
"sort": "string" }
```


The following are the responses for this request path: **`POST`:**
`/sentinel/api/v1/report/reportName`

where **`reportName`** is the id of a reportName like `TestReport`

Response


```
{
"header": {},
"data": [
{
"type": "Report Resource",
```

```
"id": "1",
"attributes": {
"CYCLEID": "Test1",
"EVENTDATE": "2025-05-14 00:00:00.0"
},
"relationships": null,
"links": null
},
{
"type": "Report Resource",
"id": "2",
"attributes": {
"CYCLEID": "Test2",
"EVENTDATE": "2025-05-14 00:00:00.0"
},
"relationships": null,
"links": null
},
{
"type": "Report Resource",
"id": "3",
"attributes": {
"CYCLEID": "Test3",
"EVENTDATE": "2025-05-14 00:00:00.0"
},
"relationships": null,
"links": null
},
.....
{
"type": "Report Resource",
"id": "1000",
"attributes": {
"CYCLEID": "Test1000",
"EVENTDATE": "2025-05-15 00:00:00.0"
},
"relationships": null,
"links": null
}
],
"included": null,
"links": {
"self": "WebDashboard/sentinel/api/v1/report/demo_sql_report1?page=1",
"next": "WebDashboard/sentinel/api/v1/report/demo_sql_report1?page=2",
"last": "WebDashboard/sentinel/api/v1/report/demo_sql_report1?page=100",
"first": null,
"prev": null
},
```

```
"meta": {
"count": 100,
"totalItems": null
}
}
```

getUserDetails

This method displays the list of all existing users.

Request type and path

`GET: /sentinel/api/v1/users`

Parameters

This method does not use additional parameters.

JSON request sample

1) By default, no input parameter required.

Request

<Sentinel_home>/SentinelWebDashboard/sentinel/api/v1/users

Responses

The following are the responses for this request path: **`GET:`** `/sentinel/api/v1/users`

Response

```
{
"status": 200,
"message": "User(s) retrieved successfully",
"data": {
"meta": {
"totalGroups": 4,
"totalUsers": 25,
"totalRoles": 4
},
"data": [
{
"relationships": {
```

```
"roles": {
"data": [
{
"name": "Admin",
"id": "1746408208",
"type": "roles"
}
]
},
"group": {
"data": {
"name": "Admin",
"id": "1695948225",
"type": "groups"
}
}
},
"id": "1695949186",
"type": "users",
"attributes": {
"userId": "Admin",
"firstName": "AdminUser",
"group": 1695948225,
"description": "Sentinel Admin",
"email": "ay@g.co",
"roles": [
1746408208
],
"lastName": "Info"
}
},
{
"relationships": {
"roles": {
"data": []
},
"group": {
"data": {
"name": "group",
"id": "1886072209",
"type": "groups"
}
}
},
"id": "993786560",
"type": "users",
"attributes": {
"userId": "sb",
```

```
"firstName": "",
"group": 1886072209,
"description": "",
"email": "",
"roles": [],
"lastName": ""
}
},
```

getGroupDetails

This method displays the list of all existing groups.

Request type and path

`GET: /sentinel/api/v1/groups`

Parameters

This method does not use additional parameters.

JSON request sample

1) By default, no input parameter required.

Request

<Sentinel_home>/SentinelWebDashboard/sentinel/api/v1/groups

Responses

The following are the responses for this request path: **`GET:`** `/sentinel/api/v1/groups`

Response

```
{
"status": 200,
"message": "Groups retrieved successfully",
"data": [
{
"id": 1695948225,
"name": "Admin",
"description": "Administration group",
"creationDate": "Mon Sep 15 10:05:09 IST 2025",
```

```
"profileId": 1746408208,
"groupVariables": []
},
{
"id": 1886072209,
"name": "group",
"description": "group",
"creationDate": "Mon Sep 15 10:05:09 IST 2025",
"profileId": 1746408208,
"groupVariables": [
{
"variable": "gr",
"type": "Boolean",
"value": "true"
},
{
"variable": "gr2",
"type": "String",
"value": "gr1"
}
]
```


CreateUser

This method create and deploy one or more users.

Request type and path

```
POST: /sentinel/api/v1/users
```

Parameters

This method requires only the standard parameters used in the body section, specifically for the User object that is to be created and deployed, with **UserId**, **Password**, and **GroupId** being mandatory fields. Modify any existing parameter values to what you need, and if a referenced parameter is unnecessary, remove it and leave its attribute value empty. To create multiple users, you may separate entries using commas at the end of each tag.

"**description**": The user description field is optional input. It can include relevant details or remain empty.

"***userId**": The User ID field is mandatory. Please provide a valid value to ensure the corresponding User ID is displayed correctly. Additionally you may include the following special characters: **_, ., and -**.

***password** : The password is mandatory field and required to corresponding user authentication.

"**email**": The email id field is optional input. It can include relevant details or remain empty.

firstName: The first name is optional input. It can include relevant details or remain empty.

"**lastName**": The last name is optional input. It can include relevant details or remain empty.

"***groupId**": By default "group Id" value is '1' displaying, kindly update the value with relevant available group id value in which you want to add user.

"**userVariables**":[

{

"**variable**": To display a value for a variable, please specify the desired input in the 'variable' parameter. Otherwise, set the all attribute value to 'empty' to display nothing.

"**type**": The value for the **Type** attribute must be one of the following: String, Boolean, Char, Integer, Date, Time, Float, or List. Please refer to the table for the correct input format.

Type Parameters	Value/Options
String	Any valid string value, such as "hello" can be used as input.
Boolean	Only Boolean values are accepted, use "true" or "false" as valid inputs.
Char	Only valid single character values are accepted such as "A".
Integer	Only valid integer values are accepted such as "0".
Date	The accepted date format is yyyy-MM-dd.
Time	The accepted time format is HH:mm:ss.
Float	Only valid float values are accepted such as "0.5".
List	Valid list inputs such as [L1, L2, L3].

"**value**": Value should be input according to selected "Type" Value.

}]

Note: To display a value for a user variable, please specify the desired input in the 'uservariables' parameter. Otherwise, set the all attribute value to 'empty' to display nothing.

JSON request sample

1) This method requires only the standard parameters used in the body section.

Request

<Sentinel_home>/SentinelWebDashboard/sentinel/api/v1/users

Request Body for Single User Creation

```
[
{
"description": "user creation",
"userId": "Mike",
"password": "Mike",
"email": "Mike@gmai.com",
"firstName": "Mike",
"lastName": "John",
"groupId": 638368368,
"userVariables": [
{
"variable": "",
"type": "",
"value": ""
}
]
}
]
```

Responses

The following are the responses for this request path: **POST:** /sentinel/api/v1/users

Response

```
[
{
"Mike": 1673153770,
"status": "SUCCESS"
}
]
```

JSON request sample

1) This method requires only the standard parameters used in the body section.

Request

<Sentinel_home>/SentinelWebDashboard/sentinel/api/v1/users

Request Body for multiple User Creation

For creating multiple user with user variable

```
[
{
"description": "string",
"userId": "AliceOne",
"password": AliceOne",
"email": "AliceOne@gmail.com",
"firstName": "Alice",
"lastName": "One",
"groupId": 1695948225,
"userVariables": [
{
"variable": "ABC",
"type": "Integer",
"value": "10"
}
]
},
{
"description": "string",
"userId": "BobTwo",
"password": "BobTwo",
"email": "BobTwo@gmail.com",
"firstName": "Bob",
"lastName": "Two",
"groupId": 1695948225,
"userVariables": [
{
"variable": "ABC",
"type": "String",
"value": "Udit"
}
]
},
{
"description": "string",
"userId": "CharlieThree",
"password": "CharlieThree",
"email": "CharlieThree@gmail.com",
"firstName": "Charlie",
"lastName": "Three",
"groupId": 1695948225,
"userVariables": [
{
"variable": "ABC",
"type": "Boolean",
"value": "True"
}
]
```

```
},
{
"description": "string",
"userId": "DinaFour",
"password": "DinaFour",
"email": "DinaFour@gmail.com",
"firstName": "Dina",
"lastName": "Four",
"groupId": 1695948225,
"userVariables": [
{
"variable": "ABC",
"type": "Char",
"value": "A"
}
]
},
{
"description": "string",
"userId": "EthanFive",
"password": "EthanFive",
"email": "EthanFive@gmail.com",
"firstName": "Ethan",
"lastName": "Five",
"groupId": 1695948225,
"userVariables": [
{
"variable": "ABC",
"type": "Date",
"value": "2025-09-12"
}
]
},
{
"description": "string",
"userId": "FionaSix",
"password": "FionaSix",
"email": "FionaSix@gmail.com",
"firstName": "Fiona",
"lastName": "Six",
"groupId": 1695948225,
"userVariables": [
{
"variable": "ABC",
"type": "Time",
"value": "10:12:12"
}
]
```

```
},
{
"description": "string",
"userId": "George",
"password": "George",
"email": "George@gmail.com",
"firstName": "George",
"lastName": "7",
"groupId": 1695948225,
"userVariables": [
{
"variable": "ABC",
"type": "Float",
"value": "10.12"
}
]
},
{
"description": "string",
"userId": "Hannah",
"password": "Hannah",
"email": "Hannah@gmail.com",
"firstName": "Hannah",
"lastName": "8",
"groupId": 1695948225,
"userVariables": [
{
"variable": "ABC",
"type": "List",
"value": "[L1,L2,L3]"
}
]
}
]
```


Responses

The following are the responses for this request path: **POST:** `/sentinel/api/v1/users`

Response


```
[
{
"AliceOne": 1874388284,
"status": "SUCCESS"
},
```

```
{
"BobTwo": 1874394049,
"status": "SUCCESS"
},
{
"CharlieThree": 1874419042,
"status": "SUCCESS"
},
{
"DinaFour": 1874423847,
"status": "SUCCESS"
},
{
"EthanFive": 1874448829,
"status": "SUCCESS"
},
{
"FionaSix": 1874453603,
"status": "SUCCESS"
},
{
"George": 1874478560,
"status": "SUCCESS"
},
{
"Hannah": 1874483306,
"status": "SUCCESS"
}
]
```

Known limitations 2

As of Sentinel 4.2.0 SP33, there are following known limitation of the Open Sentinel API:

- Open Sentinel API cannot be deactivated. It is always available when Sentinel WebDashboard is installed.
- Simultaneously utilizing same user with both "Sentinel Web Dashboard" and "Open Sentinel API" via third-party applications is not supported.
- At present, Open Sentinel API only supporting filters with SQL DD. In the near future, we plan to provide support for the remaining data dictionaries.
- Currently reports with cross table are not supported.
- The new Web Administration Open API is only accessible when Sentinel WebDashboard is installed.