API Guide

FortiSOAR 7.4.3

TABLE OF CONTENTS

Change Log

Date	Change Description
2024-02-21	Updated the URL for the Authenticate API in the "API Methods" chapter.
2023-12-01	Initial release of 7.4.3

Overview

The FortiSOAR™ application uses a Representational State Transfer (RESTful) API with a Service Oriented Architecture (SOA) to allow for scalability, flexibility, and ease of use. The characteristics of this design include:

- Simplicity of interface
- Scalability, especially when combined within the SOA microservices concepts
- Maintainability, while the application code may change underneath the API, the API definition may remain in place
- Performance, component interactions dominate the user-perceived performance and efficiency

The API gives you access to the primary data available to the core system, which are also consumed by the Angular.js client for front end interaction.

FortiSOAR uses the Hydra Specification for implementing the APIs.

While direct access to the module data models and endpoints is provided, we recommend that you consider the Playbooks system as the primary method for working to get data input and output from the FortiSOAR platform. The Playbook system makes development simpler, easier, and more flexible for both ingest and export of data via RESTful methods without requiring debugging or maintenance when new releases are available. However, there may be cases where the Playbooks system does not allow for the required functionality, and custom development on the direct API is needed.

Concepts

JSON-LD

The API uses the JSON for Linking Data (JSON-LD) format to present information. JSON-LD empowers humans and machines to read and write across the core API without pre-existing knowledge of the API structure, other than the root API endpoint. JSON-LD uses embedded links to determine the Context of the API endpoints quickly and easily in a human-readable format.

Example:
API endpoints for `"@context": "/api/3/contexts/"` and `"@id": "/api/3/"` are standard pieces of the JSON-LD format.

Hydra Vocab

Additionally, the JSON-LD structure is combined with the Hydra API definition, which applies a standard Core Vocabulary to all data within the API. The Hydra definition is a combination of the Core Vocabulary and the JSON-LD structure to completely define the entirety of the API within itself, making the API standardized and discoverable at the same time.

> Any API information described as `"@vocab"` is a part of the core Hydra definition.

UUIDs

Universally unique identifiers (UUIDs) are used throughout the system to ensure all records have a unique identifier. The canonical displayed versions are 36 characters using hexadecimal text with inserted hyphen characters in the following format: `123e4567-e89b-12d3-a456-426655440000`

When creating new records directly in the API, you may optionally include a generated UUID if you wish to retain that UUID in an external system for reference. This must be a differentiated UUID from any other record in the system, or it will fail.

If you do not include a UUID during a POST, a UUID will be generated for you. This is the recommended method as the response for the POST will contain the IRI with included UUID that you may retrieve and save.

IRIs

All foreign key references use International Resource Identifiers (IRIs) to link records within the system. The format for this is essentially identical to a URI, forming a relative link within the application, though there are differences in the formats. The identifier uses a UUID associated with the record appended to the record location from the API root.

`/api/3/alerts/{uuid}`

IRIs will be seen in the API typically at the top of a record using the key `"@id"`.

> The IRI is automatically generated when inserting the record and is not intended to be used during any POST activities. POST activities should use the `"uuid":` identifier with a plain 36-character UUID.

Organization

The REST API is organized around the core Modules defined within the platform. Note that most Modules are defined on a 1:1 basis with a database table, though this is not always the case. The root API may be found at `https://FortiSOAR_ROOT.URL/api/3`, but can only be accessed with the proper authentication. See the `Authentication API` section in the *API Methods* chapter for more information.

All Modules and models are made available in the core API, Modules representing a specific data table accessible within the UI. New Modules may be added via the Module Editor in the Administration menu, and every new Module will add a new API endpoint for accessing the records within that Module.

Authentication

Authentication methods for using the API are available via two methods:

1. **Hash-based message authentication code (HMAC)** - this method involves computing a specific cryptographic hash via a cryptographic key within the parameters specified from FortiSOAR. This is the required method for permanent API access. See the HMAC Authentication section for more details.
   Note: For HMAC authentication the timestamp must be in UTC format.
2. **Token-based authentication** - this method is a temporary access mechanism by using a SHA-256 token, available within the browser resources of the UI. This token expires on a periodic basis and must be refreshed, however, it is useful for temporary access when doing specific API actions during development where generating an HMAC key is not possible, e.g., REST clients and CURL methods.
   The token must be placed in the Header with the `Authorization` key and the format for identifying the user a `Bearer %token%`

FortiSOAR utilizes a sessionless security model, meaning the server does not track individual user sessions as objects. Users must be authenticated with a unique token that has a limited lifespan, typically configured at 30 minutes though that may be adjusted. For more information, see the `Authentication API` section in the *API Methods* chapter.

Using the HMAC method does not require a session because each message is individually signed with the cryptographic key, but this must be computed, and the server clocks must be in sync for proper operation.

> Basic authentication and No authentication webhooks are allowed in the context of the Custom API Endpoint for Playbooks. This is purposeful to reduce the overall risk profile of allowing for less secure methods on the core API.

Authorization

Every API action is reviewed against user authorization privileges. These are determined by your Team membership and Role assignment as described in the "Administration Guide.

> We recommend that you always use a limited privilege set scoped to the desired operations to ensure both security and guard against potential data loss due to accidental actions.

Operation

Exercising the API is possible via any valid REST client, such as Postman, or using CURL combined with a user token. All valid CRUD operations allowed via your user privileges may be invoked, but not all endpoints support all HTTP methods.

Supported HTTP Methods

- **GET [Read]**
- **POST [Create]**
- **PUT [Update]**
- **DELETE [Delete]**

URI Encoding

The most common format for traversing the API involves using the Module names in the following format.

```
/api/3/{module name}
```

An example endpoint for Alerts is as follows:

```
/api/3/alerts
```

The body response will include a list of the Alert records found within the database.

```
{
    "@context": "/api/3/contexts/Alert",
    "@id": "/api/3/alerts",
    "@type": "hydra:PagedCollection",
    "hydra:nextPage": "/api/3/alerts?%24page=2",
    "hydra:totalItems": 1,
    "hydra:itemsPerPage": 30,
    "hydra:firstPage": "/api/3/alerts",
    "hydra:lastPage": "/api/3/alerts?%24page=2",
    "hydra:member": [
        {
        "@id": "/api/3/alerts/028b37fa-bb35-4e3b-8afb-a3274a8cb343",
        "@type": "Alert",
        "name": "Malicious Attachment - Malware.Binary.Vbs",
        "sourceId": null,
        "source": "FireEye EX - Email MPS",
        "origin": "",
        "dueDate": null,
        "description": null,
        "type": null,
        "severity": null,
        "status": null,
        "assignedTo": null,
        "createUser": {
        "@id": "/api/3/appliances/432ba9fe-0955-4379-9177-68b0d87e8caf",
        "@type": "Appliance",
        "name": "Workflow",
        "userId": "7c23b46e-5b84-47c0-8f7f-fe0498156c81",
        "createUser": null,
        "createDate": null,
        "modifyUser": "/api/3/people/bcb46d5a-0ad8-4480-ac68-9ebc28502a30",
        "modifyDate": 1457479171,
        "id": 502,
        "@settings": "/api/3/user_settings/432ba9fe-0955-4379-9177-68b0d87e8caf"
        }]
    "hydra:search": {
```

```
    "@type": "hydra:IriTemplate",
    "hydra:template": "/api/3/alerts{?}",
    "hydra:variableRepresentation": "BasicRepresentation",
    "hydra:mapping": []
    }
}
```

Syntax

FortiSOAR uses the CamelCase format for all data labels, e.g., keys for the key-value pairs in the JSON structure. CamelCase is the practice of writing a compound word or phrase structures where the first letter of the first word is lowercase and subsequent first letters of words following are capitalized.

	We recommend you also use CamelCase format during your development to ensure consistency and readability.

Examples

```
"hydra:variableRepresentation"
"sourceId"
"dueDate"
```

API Endpoints Reference

This gives a summary of the API endpoints available within FortiSOAR organized via category.

Category	Description
Modules	General access for CRUD operations on records within the Modules defined in the system database
Playbooks	Access to both Playbook models, Step types (core system), Steps, and Custom Endpoints (Custom API Endpoint Triggers)
Configuration	General management of the configuration of the system, including Users / Appliances (actors), View Templates, and Module / Model definitions

Query Parameters

Most endpoints allow for query parameters to be appended to the URL for scoping the specific data set being returned. Parameters are used to filter results from the query and change the result format.

Syntax

1. The beginning of the query string is denoted by a `?` appended to the end of a valid endpoint URL.
2. Operational parameter values should start with a `$`.
3. Filter values should not have a `$`.
4. Parameters specify the specific value using `=`.
5. Parameters are chained using the `&` character.

Sample URL

```
https://FortiSOAR.ROOT.URL/api/3/{collection}/{uuid}?parameter1=value&parameter2=value
```

Operations

The standard query parameters are given in the following table.

Parameter	Valid Values	Action
$relationships	true, yes, false (default)	Show or hide full field and value context for related records
$export	true, yes, false (default)	Display record in exportable/importable JSON format, e.g., `"@id"` is turned from an IRI to a simple `"uuid"` value

{fieldName}	Any valid format for field type with url-encoded for special characters	Filter results by `fieldName` using the equivalence operator. A `fieldName` can contain double underscores to specify a key within an object. For example, `status__itemValue` or `assignedToPerson__firstname`.
{fieldName}${operator}	Any valid format for field type with url-encoded for special characters	Filter results by `fieldName` using any operator. Operators that are supported are found in the *Query API Reference* chapter. A `fieldName` can contain double underscores to specify a key within an object.
$orderby	Any valid `{fieldName}`, optional – `{fieldName}` operand	Sort the returned data set by the values in a specific field, may be chained and accepts `-value` for inverse ordering
$limit	An integer between 1 and 214748364	Maximum number of records to return in a query (default is 30)

Examples

```
/api/3/alerts?name=Full Alert Name
/api/3/alerts?name$like=%partial name%
/api/3/alerts?eventCount$gte=10&eventCount$lt=20
/api/3/alerts?status__itemValue=Open
/api/3/tasks?assignedToPerson__firstname$like=%doe%
/api/3/alerts?status__itemValue$in=Open |Resolved|In Progress
/api/3/alerts?$limit=100&$orderby=-createDate&$relationships=true
```

Module Endpoints

Module endpoints are grouped here by the Component to which they belong. At the root of the API, the models are not organized by Component. The component definition only exists within the system view metadata for defining the navigation.

An important concept to grasp is that the Modules within the UI generally refer to a 1:1 relationship with a table. Beyond the UI, many of the Modules in the Role Editor are actually composed of multiple data models within the API. This section refers to the Modules that have a 1:1 data model relationship.

There are 3 optional endpoint formats defined within all modules. Each of these endpoints supports a specific set of standard operations.

Module Root

```
/api/3/{plural_module_name}
```

The Module Root displays a paginated listing of all records within that module. The default sorting is by last modified date.

```
/api/3/{plural_module_name}?{$filter_query}
```

The Module Root may be filtered to a specific set of criteria as opposed to generally listing all models within the database.

Supported Operations

All Module Root endpoints support the following operations.

Operation	Description	Formats
GET	Allows you to read the list of records	-
POST	Insert a new record of the specified module type	JSON
PUT	Update an existing record, requires "@id"	JSON
DELETE	Delete an existing record, requires, requires "@id"	-

Module Records

```
/api/3/{plural_module_name}/{record_uuid}
```

Supported Operations

All Module Record endpoints support the following operations.

Operation	Description	Formats
GET	Allows you to read the list of records	-

Playbook Endpoints

Custom Endpoints (Custom API Endpoint Trigger)

Endpoints Available

```
https://FortiSOAR.ROOT.URL/api/triggers/1/{name}
OR
https://FortiSOAR.ROOT.URL/api/triggers/1/deferred/{name}
```

These are user-defined endpoints generated based on the specified arbitrary name included in the Playbook trigger.
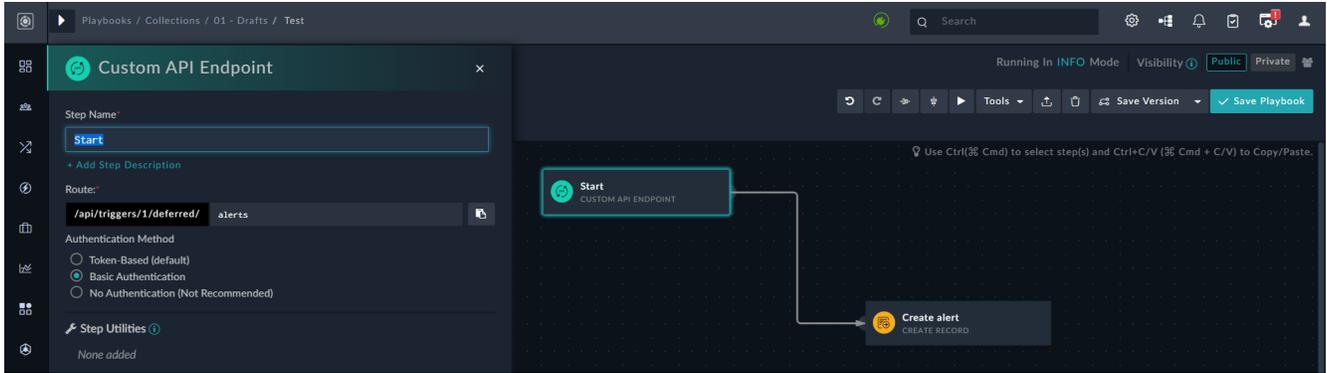
The `deferred` endpoints allow for alternate authentication means, i.e., no HMAC (token) requirement, for external systems that may only offer Basic or non-authenticated webhooks (No Authentication).

Configuration Endpoints

The Configuration endpoints primarily govern system and user interaction. Generally, it should not be necessary to use these endpoints outside of the UI, but they are provided here as a reference.

Model Metadatas

⚠️ Modifications to the Model Metadatas and Attributes are not recommended due to the potential for database manipulation leading to a potentially difficult recovery.

`Model Metadata` describes every piece of customizable data models in our system. We have two pieces of metadata: `Model` and `Attribute`. `Models` are comprised of any number of `Attributes`. Attributes can be of any type, ranging from any JSON primitive type to relationships with existing entities. These relationships can be of type 'toMany' or 'toOne'.

ModelMetadata: `/api/3/model_metadatas`

- Type [`type`]
  Metadata type, JSON primitive for attributes or internal types for models. See `type` for `AttributeMetadata` for valid JSON primitives.
- Parent Type [`parentType`]
  The type of the parent, if set to `null`, will default to the `BaseEntity` class.
- Table Name [`tableName`]
  The table name for ORM, this will be the table that is created in the database.
- Ownable [`ownable`]
  Determines if this class should have the ownable fields, user ownable and team ownable.
- Trackable [`trackable`]
  Determines if this module should have the trackable fields (`createDate`, `createUser`, `modifyDate`, `modifyUser`)

- Indexable [`indexable`]
  Determines if this class should have the indexable fields (`id`). This gives the object a global sequence id field.
- Attributes [`attributes`]
  Key-value list of the attributes for this class. These attributes can be described here: `/api/3/attrib_model_metadatas`
- Display Template [`displayTemplate`]
  Template for the display name of record as an AngularJS template.
- Descriptions [`descriptions`]
  Map of description type => description, example:

  ```
  {
      "singular" : "Alert",
      "plural" : "Alerts"
  }
  ```

AttributeMetadata: `/api/3/attrib_model_metadatas`

- Id [`id`]
  The unique object id for this attribute, will be auto generated if not specified.
- Type [`type`]
  Metadata type, JSON primitive for attributes or iri/internal types for models. Supported primitives are:
  `integer,`
  `string,`
  `id,`
  `boolean,`
  `number,`
  `null`
  `object,`
  `array`
  Note: Adding an array field type is currently not supported through the FortiSOAR UI.
  `object` and `array` are special types. See the *Extended Data* section below.
  Tip: `null` is supported in theory but might not make sense to use.
- Name [`name`] (**required**, **unique**) The name of the attribute.
  Note: DO NOT use underscores in the field name. Must be Camel-Case.
- Length [`length`]
  The length of the attribute. For anything less than 255, it will created in the SQL database as a VARCHAR with a defined length.
- FormType [`formType`]
  The type of form to use to render this attribute. Supported types are:
  `integer,`
  `text,`
  `textarea,`
  `checkbox,`
  `manyToMany,`
  `datetime,`
  `picklist,`
  `lookup,`
  `oneToMany,`
  `file,`
  `manyToOne,`
  `richtext,`
  `phone,`

```
email,
json
```

- Data Source [dataSource]
  The source of the data, not defined explicitly. An example of picklist lookups:

```
{
    "module" : "picklists",
    "query": {
        "logic": "AND",
        "filters": [
            {
                "field": "listName__name",
                "operator": "eq",
                "value": "Criticality"
            }
        ],
        "sort": [
            {
                "field": "orderIndex",
                "direction": "ASC"
            }
        ]
    }
}
```

  An example of a relationship for all items in a collection, query is null because no query should be made to limit the lookup:

```
{
    "module" : "alerts",
    "query" : null
}
```

- Searchable [searchable]
  Determines if this attribute should be searchable within a grid in the FortiSOAR UI. This does not affect record indexing in ElasticSearch. The field will yet be displayed in the global search results in the UI even if it is not marked as searchable.
- Grid Column [gridColumn]
  Determines if this attribute should show up in a grid by default.
- Mapped By [mappedBy]
  Field name on the referenced type, if it is a collection and not the owner of a relationship.
- Inversed By [inversedBy] (**validation enforced**)
  Field name on the referenced type, if it is a collection and is the owner.
- Owns Relationship [ownsRelationship] (**validation enforced**)
  Whether this model owns the relationships, if is collection
- Validation [validation]
  Validation object, a list of validation attributes with boolean values or condition sets. Examples:
  { "required" : true }
  { "required" : false }
  Example of conditional validation: Only require resolved if status equals Resolved, only

```
{
    "required": [
        {
            "logic": "AND",
            "result": true,
```

```
                        "filters": [
                            {
                                "field": "status",
                                "operator": "eq",
                                "value": "/api/3/picklists/97e9bc03-5a4c-43c6-b3a3-47422b42d288"
                            }
                        ]
                    }
                ]
            }
```

- Visibility [`visibility`]
  Determines if this attribute should be visible during data entry.
  This can be the following:
  `true`, the default, will always be shown.
  `false`, will not show up in the application.
  Query object, see above, will be visible if the condition matches.

- Readable [`readable`]
  Determines if this attribute should be returned during READ operations.

- Writeable [`writeable`]
  Determines if this attribute should be modified during UPDATE operations.

- Identifier [`identifier`]
  Determines if this attribute is a unique identifier for this class.

- Unique [`unique`]
  Determines if the attribute is unique.

- Display Name [`displayName`]
  Display name as a template using field names, example:

```
{
    "displayName" : "{{ sourceId }}"
}
```

- Descriptions [`descriptions`]
  Map of description type => description, example:

```
{
    "singular" : "Source ID"
}
```

Examples

For the following examples, we will use `alerts`.

Viewing Model Metadata

To view a model, use `GET` on `/api/3/model_metadatas/7ab10952-b2ea-4910-9762-eb685bdd29eb`. You should receive a result like this:

```
{
  "@context": "/api/3/contexts/ModelMetadata",
  "@id": "/api/3/model_metadatas/7ab10952-b2ea-4910-9762-eb685bdd29eb",
  "@type": "ModelMetadata",
  "parentType": "/api/3/model_metadatas/base_entities",
  "tableName": "alerts",
```

```
    "ownable": true,
    "trackable": true,
    "indexable": true,
    "displayName": "{{ name }}",
    "descriptions": {
      "singular": "Alert",
      "plural": "Alerts"
    }
}
```

Model Attributes

All models will have attributes to describe the fields on the json. These can be retrieved by using:

```
GET /api/3/model_metadatas/<uuid>?$relationships=true
```

```
{
    "@id": "/api/3/model_metadatas/08854338-6440-4b66-a07f-9788086fa088",
    "@type": "ModelMetadata",
    "attributes": [
        {
            "@id": "/api/3/attrib_model_metadata/8582cab2-ed5a-4daa-abaa-0277cba104e9",
            "@type": "AttribModelMetadata",
            "sattrib": "/api/3/model_metadatas/08854338-6440-4b66-a07f-9788086fa088",
            "type": "string",
            "name": "firstname",
            "length": 255,
            "formType": "text",
            "dataSource": [],
            "searchable": true,
            "system": true,
            "encrypted": false,
            "gridColumn": true,
            "collection": false,
            "inversedField": null,
            "ownsRelationship": false,
            "validation": {
                "required": true,
                "minlength": 0,
                "maxlength": 255
            },
            "bulkAction": {
                "allow": false
            },
            "defaultValue": "",
            "dataSourceFilters": [],
            "tooltip": "",
            "visibility": true,
            "readable": true,
            "writeable": true,
            "identifier": false,
            "unique": false,
            "peerReplicable": true,
            "displayName": "{{ firstname }}",
            "descriptions": {
                "singular": "First Name"
            }
```

```
            }
        ],
        "type": "people",
        "parentType": "actors",
        "tableName": "actors",
        "ownable": false,
        "trackable": true,
        "peerReplicable": false,
        "defaultSort": [
            {
                "field": "firstname",
                "direction": "ASC"
            },
            {
                "field": "lastname",
                "direction": "ASC"
            }
        ],
        "indexable": true,
        "writable": true,
        "displayName": "{{ firstname }} {{ lastname }}",
        "descriptions": {
            "singular": "Person",
            "plural": "People"
        }
    }
}
```

Exporting a Model

To export a model, use `GET` on `/api/3/model_metadatas/<uuid>?$relationships=true&$export=true`. You should receive a result like this:

```
{
  "@type": "ModelMetadata",
  "uuid": "7ab10952-b2ea-4910-9762-eb685bdd29eb",
  "type": "alerts",
  "tableName": "alerts",
  "attributes": [
    {
      "@type": "AttributeMetadata",
      "uuid": "16c855d3-29b4-4d29-90a8-8f322b556067",
      "name": "name",
      "length": 255,
      "searchable": true,
      "gridColumn": true,
      "validation": {
        "required": true
      },
      "displayName": "{{ name }}",
      "descriptions": {
        "singular": "Name"
      }
    },
    ...
```

```
    ]
}
```

Updating Model Metadata

To update a model, first, you should GET the model you would like to update. Then, use PUT with the fields you want to change:

```
PUT /api/3/staging_model_metadatas/7ab10952-b2ea-4910-9762-eb685bdd29eb

{
      "ownable": true
}
```

To update attributes on a model, first, you should GET the model. Then use PUT on the attributes field with the modifications. You need to post all the attributes as a whole.

```
PUT /api/3/staging_model_metadatas/7ab10952-b2ea-4910-9762-eb685bdd29eb

{
  "attributes": [
    {
      /* Existing with @id */
    },
    {
      /* New without @id */
    }
  ]
}
```

You can update an attribute. For example, going to /api/3/attribute_metadatas/56d50d6a5a4ef3f3348b4574 will return a single metadata.

The /api/3/staging_model_metadatas API also manages replication configuration for modules that have 'peerReplicable' set to "True". In addition, the 'replicationFilter' field contains the replication condition specified for each module.

Creating New Models

To create a new model, use the same syntax as you have used for updating the models; except use POST and specify a type and parentType.

```
POST /api/3/model_metadatas

{
      "type": "malware",
    "parentType": "/api/3/model_metadatas/base_entities",
    "descriptions": {
        "singular": "Malware",
        "plural": "Malware"
    },
    "tableName": "malware"
}
```

The plural description will be used as the Module description. TableName is required for Postgres implementations of this model.

View Template

Create List View

```
POST /api/3/system_view_templates

{
        "id": "modules/malwares-list",
        "type": "rows",
        "config": {
                "rows": [{
                        "columns": [{
                                "widgets": [{
                                        "type": "grid",
                                        "config": []
                                }]
                        }]
                }]
        }
}
```

Create Detail View

```
POST /api/3/system_view_templates

{
        "id": "modules/malwares/detail",
        "type": "rows",
        "config": {
                "rows": [{
                        "columns": [{
                                "widgets": [{
                                        "type": "form",
                                        "config": []
                                }]
                        }]
                }, {
                        "columns": [{
                                "widgets": [{
                                        "type": "relationship.subtab",
                                        "config": []
                                }]
                        }]
                }]
        }
}
```

Extended Data

The attribute types that are type: `object` or `array` are often referred to as: 'Extended Data'. These are schema-less json blobs that have minimal input validation.

Attribute metadata for these types will look like this:

```
[
    {
        "@type": "AttributeMetadata",
        "uuid": "60e00773-2de6-44aa-8af0-174e4fbdec17",
        "name": "extendedData",
        "type": "object",
        "formType": "object",
        "displayName": "{{ extendedData }}",
        "descriptions": {
            "singular": "Extended Data"
        }
    },
    {
        "@type": "AttributeMetadata",
        "uuid": "338040a1-6965-4e56-b71d-80cc83ecbeb7",
        "name": "extendedDatas",
        "type": "array",
        "formType": "array",
        "displayName": "{{ extendedDatas }}",
        "descriptions": {
            "singular": "Extended Data (Array)"
        }
    }
]
```

Examples of Valid Data

```
{
    "extendedData": {
        "test": 1234,
        "key2": [
            "randomdata"
        ]
    }
}

{
    "extendedDatas": [
        1234,
        {
            "nested": "object"
        }
    ]
}
```

Examples of Bad Data

```
{
    "extendedData": [
        "array",
        "on object",
        "is bad"
    ]
}
```

```
{
    "extendedDatas": {
        "do not": "post object to array"
    }
}
```

Indexing of Extended Data

Because of the limitations on ElasticSearch, extended data needs to be transformed into a string before it is indexed. Searches performed on this data will act like a traditional text search.

System Update

After making changes to the metadata, it is time to publish. If you have the UPDATE permission to the Application module, make a PUT command to `/api/publish`. You will get a response like this:

```
{
    "@type": "Publish",
    "status": "started"
}
```

This will indicate that a system update has begun. This will make every request to the API return a 503 until the publish is complete. The progress will look something like this:

```
{
  "@type": "SystemUpdate",
  "code": 503,
  "message": "Clearing cache",
  "progressPercent": 22,
  "startTime": 1460678007
}
```

Modules List

- Route: modules/{{ module }}
- Template: modules/{{ module }}/list
- State: main.modules.list
- Parameters:
    - module: name of module

Model-dependent page that will be able to utilize the state parameters to build widgets.

Simple Example

```
{
  "id": "modules/incidents/list",
  "type": "rows",
  "config": {
    "rows": [{
      "columns": [{
        "widgets": [{
```

```
            "type": "grid"
          }]
        }]
      }]
    }
  }
```

Modules Detail

- Route: modules/{{ module }}/{{ id }}
- Template: modules/{{ module }}/detail
- State: main.modulesDetail
- Parameters:
  - module: name of the module
  - id: id of the given record

Model-dependent page that will be able to utilize the state parameters to build widgets.

Simple Example

```
{
  "id": "modules/incidents/detail",
  "type": "rows",
  "config": {
    "rows": [{
      "columns": [{
        "widgets": [{
          "type": "form"
        }]
      }]
    }, {
      "columns": [{
        "widgets": [{
          "type": "relationship.subtab",
          "config": []
        }]
      }]
    }]
  }
}
```

Modules Form

- Route: modules/{{ module }}/add or modules/{{ module }}/edit/{{ id }}
- Template: modules/{{ module }}/form
- State: main.modulesAdd or main.modulesEdit
- Parameters:
  - module: name of the module
  - id: id of the given record (on edit only)

Module edit and module add both use the form template to load in a template that both modes can share to display fields and a form to the user. Model-dependent page that will be able to utilize the state parameters to build widgets.

Example

```
{
  "id": "modules/incidents/form",
  "type": "form",
  "config": {
    "rows": [{
      "columns": [{
        "widgets": [{
          "config": {
            "rows": [{
              "columns": [{
                "fields": [
                  "name"
                ]
              }]
            }],
            "size": "large"
          },
          "type": "formGroup"
        }, {
          "config": {
            "rows": [{
              "columns": [{
                "fields": [
                  "description"
                ]
              }]
            }]
          },
          "type": "formGroup"
        }, {
          "config": {
            "rows": [{
              "columns": [{
                "fields": [
                  "phase",
                  "status"
                ]
              }]
            }],
            "title": "Summary"
          },
          "type": "formGroup"
        }, {
          "config": {
            "rows": [{
              "columns": [{
                "fields": [
                  "resolution"
                ]
              }]
```

```
          }],
          "title": "Resolved"
        },
        "type": "formGroup"
      }, {
        "config": {
          "rows": [{
            "columns": [{
              "fields": [
                "confirmation",
                "confidence",
                "severity",
                "category",
                "source",
                "originPoint",
                "incidentLead"
              ]
            }]
          }],
          "title": "Details"
        },
        "type": "formGroup"
      }, {
        "config": {
          "rows": [{
            "columns": [{
              "fields": [
                "dateOfIncident",
                "discoveredOn",
                "dwellTime",
                "containmentTime",
                "recoveryTime"
              ]
            }]
          }],
          "title": "Dates"
        },
        "type": "formGroup"
      }]
    }]
  }]
  }
}
```

Application

Template: app

The application template is a unique widget that is loaded when you login to exo. This template describes header and navigation details.

Simple Example

```
{
  "id": "app",
  "type": "app",
  "config": {
    "header": {
    },
    "navigation": [{
      "title": "Home",
      "icon": "fa fa-home",
      "require": [],
      "state": {
        "name": "main.dashboard",
        "parameters": []
      }
    }, {
      "title": "Vulnerability Management",
      "icon": "fa fa-bug",
      "items": [{
        "title": "Vulnerabilities",
        "require": {
          "module": "vulnerabilities",
          "action": "canRead"
        },
        "state": {
          "name": "main.modules.list",
          "parameters": {
            "module": "vulnerabilities"
          }
        }
      }, {
        "title": "Assets",
        "require": {
          "module": "assets",
          "action": "canRead"
        },
        "state": {
          "name": "main.modules.list",
          "parameters": {
            "module": "assets"
          }
        }
      }]
    }]
  }
}
```

Header

Header property is the collection of the header component details.

Navigation

Navigation property is an array of links to build the left-hand navigation bar.

Links

A single link can be created like below.

```
{
  "title": "Home", // Title of the page
  "icon": "fa fa-home", // Icon used for the link
  "require": {  // Restriction of the link. or collection of links
    "module": "vulnerabilities", // What module to restrict on
    "action": "canRead" // Restrited action (defaults to canRead).
  },
  "state": { // State and it's parameters
    "name": "main.dashboard",
    "parameters": []
  }
}
```

A collection of links will look like the following:

```
{
  "title": "Vulnerability Management", // Title of collection
  "icon": "fa fa-bug", // Icon used for the collection
  "items": [{ // array of links (same as single link)
    "title": "Vulnerabilities",
    "require": {
      "module": "vulnerabilities",
      "action": "canRead"
    },
    "state": {
      "name": "main.modules.list",
      "parameters": {
        "module": "vulnerabilities"
      }
    }
  }, {
    "title": "Assets",
    "require": {
      "module": "assets",
      "action": "canRead"
    },
    "state": {
      "name": "main.modules.list",
      "parameters": {
        "module": "assets"
      }
    }
  }]
}
```

HMAC Authentication

To get public/private key, you need to create the appliance in the system and the copy keys from that system. For more information, see the `Appliances` section in the *Security Management* chapter of the "Administration Guide."

How FortiSOAR Authorizes Payloads with HMAC

To successfully authenticate our payloads, we are required to get a fingerprint as part of the request and compare it locally by recreating the fingerprint. This is subjective and can be defined by the HMAC provider (FortiSOAR). FortiSOAR fingerprinting will be based on hashing the private key against the identifier (NOT THE ACTUAL PAYLOAD).

How to create an Identifier

The FortiSOAR identifier is created in such a way that some pieces will be gathered from the request and others will have to be provided by the requester. Our current identifier simply is: `ALGO.VERB.TIMESTAMP.FULL_URI.HASHED_PAYLOAD`

The identifier units are separated with periods. Here is a brief explanation of each unit:

- ALGO: The Algorithm the hashing is being done with, for a full list of hashing algorithms available in PHP you can use the hash_algos function.
- VERB: The current request action, GET, POST, PUT, PATCH, DELETE
- TIMESTAMP: The timestamp the fingerprint is created. This is also sent as part of the request.
  Note: Timestamp must be in UTC format.
- FULL_URI: The full URI of the request, including any parameters
- HASHED_PAYLOAD: The string that is to be fingerprinted. Hashed with the requested ALGO

Once the above is created the fingerprint is created with `hash_hmac(algo, identifier, private_key, false)`. The fingerprint is expected to be represented as a string as opposed to raw binary.

PHP Example

```
$privateKey = "our_private_key";
$publicKey = "our_public_key"; // used in the example following this

  $testAlgo = "sha256";
    $testVerb = "GET";
    $testTimestampObj = new \DateTime();
    $testTimestamp = $testTimestampObj->format('Y-m-d H:i:s');
    $testFullUri = "http://example.com/api/1/data/1";

  $testPayload = "This is a test string"; // this could be

our_public_key on GET requests
    $hashedPayload = hash($testAlgo,$testPayload,false);
```

```
            $rawFingerprint =
                $testAlgo.'.'.
                $testVerb.'.'.
                $testTimestamp.'.'.
                $testFullUri.'.'.
                $hashedPayload;

        $hashedFingerprint =
            hash_hmac(
                $testAlgo,
                $rawFingerprint,
                $privateKey,
                false
            );
```

Python Example

```python
import base64
import hashlib
import hmac
from datetime import datetime
import requests
import json


DEFAULT_ALGORITHM = "sha256"

# update below with proper fqhn or ip, and update the route name per the playbook start step
settings
API_ROUTE = "https://<fqhn-or-ip>/api/triggers/1/<route>"

with open(
    "APPLIANCE_PUBLIC_KEY",
    "r",
) as public_key_file:
    public_key = public_key_file.read().strip()
with open(
    "APPLIANCE_PRIVATE_KEY",
    "r",
) as private_key_file:
    private_key = private_key_file.read().strip()

def generate_hmac(method, full_uri, payload, private_key, public_key):
    if method == "GET":
        payload = public_key
    timestamp = datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S")
    payload = payload if type(payload) is bytes else payload.encode()
    digest_method = hashlib.new(DEFAULT_ALGORITHM)
    digest_method.update(payload)

    hashed_payload = digest_method.hexdigest()
    raw_fingerprint = "{0}.{1}.{2}.{3}.{4}".format(
        DEFAULT_ALGORITHM, method, timestamp, full_uri, hashed_payload
    )
    hashed = hmac.new(private_key.encode(), raw_fingerprint.encode(), hashlib.sha256)
    hashed_fingerprint = hashed.hexdigest()
```

```
    header = base64.b64encode(
        "{0};{1};{2};{3}".format(
            DEFAULT_ALGORITHM, timestamp, public_key, hashed_fingerprint
        ).encode()
    )
    return "CS {}".format(header.decode())

method = "POST"
payload = {"data": "test"}
auth_header = generate_hmac(
    method, API_ROUTE, json.dumps(payload), private_key, public_key
)
headers = {"Authorization": auth_header}
try:
    req = requests.request(
        method, API_ROUTE, data=json.dumps(payload), headers=headers, verify=False
    )
    print(req.json())
except Exception as e:
    print(e)
```

Using the Signature

After creating our identifier, we can send the payload to our API. However, the fingerprint alone is not enough. We'll also need to send our public key, the timestamp used to hash our fingerprint, and the algorithm being used in the request. These pieces should be concatenated using semicolons and included in the authorization header base64 encoded.

PHP Example

```
$header =
base64_encode($testAlgo.';'.$testTimestamp.';'.$publicKey.';'.$hashedFinge
rprint);

curl_setopt($ch, CURLOPT_HTTPHEADER, array(
 "Authorization: CS " . $header,
 "X-CS-Data: " . $testPayload
));
```

Python Example

```
import requests
import json
auth_header = generate_hmac(method,full_url, json.dumps(payload), private_key, public_key)
headers =
{
  'Authorization': auth_header
}
try:
  req = requests.request(method, full_url, data=payload, headers=headers, verify=False)
  print (req)
except Exception as e:
  print(e)
```

If the fingerprint is authorized the request will be delivered to the API; otherwise, a 401 error will be returned.

API Methods

CRUD API

Authentication API

Authenticate

This API is used to generate a JWT authentication token for FortiSOAR authentication using username and password. It returns a JSON response containing the JWT token which you can then use to call other FortiSOAR APIs by passing it into the Authorization header. For example, `Authorization: "Bearer {Token}".`

Request

```
METHOD: POST
URL: https://{HOSTNAME}/auth/authenticate
BODY:
  {
      "credentials":
    {
      "loginid": "{{username}}",
      "password":"{{password}}"
    }
  }
```

Response

```
{
      token:  "VALID JWT TOKEN"
}
```

Two-Factor Authentication (2FA) Methods

You can use the following APIs to work with 2FA and also add and integrate a new two-factor authentication (2FA) or N-Factor Authentication (NFA) method with FortiSOAR.

Permissions Required

Action Name	Endpoint	Method	Permissions
Enable an existing NFA method	/api/auth/nfamethod	PUT	security.update
Delete an existing NFA method	/api/auth/nfamethod	DELETE	security.delete
Get a list of all NFA methods	/api/auth/nfamethod	GET	application.read
Add or Update global configurations in an existing NFA	/api/auth/nfaconfig	POST	security.update

method			
Get global configurations for the specified NFA method	/api/auth/nfaconfig	GET	security.read
Configure an NFA User	/api/auth/nfauser	POST	application.read
Activate an NFA User	/api/auth/nfauser	PUT	application.read
Get details of existing NFA users	/api/auth/nfauser	GET	application.read
Create a new NFA method	/api/auth/nfamethod	POST	security.update

Get a list of all NFA methods

Use this API to retrieve a list of all NFA methods integrated with your FortiSOAR system. This API is used in the "Authentication/User" section in the FortiSOAR UI.

Request

```
METHOD: GET
URL: https://{HOSTNAME}/api/auth/nfamethod
```

Response

```
{
    "hydra:totalItems": 2,
    "hydra:member": [
        {
            "nfa_id": 8,
            "service_name": "Telesign",
            "available": false,
            "title": "Telesign",
            "traits": [
                "trait_configure_self",
                "trait_custom_message",
                "trait_generate_code",
                "trait_health_check"
            ],
            "config_schema": {
                "user": {
                    "inputFields": [
                        {
                            "name": "phone",
                            "type": "phone",
                            "title": "Phone number",
                            "tooltip": "Phone number to be configured to receive OTP
messages",
                            "visible": "True",
                            "editable": "True",
                            "required": "True",
                            "description": "",
                            "placeholder": ""
                        },
                        {
                            "name": "mode",
                            "type": "select",
```

```
                        "title": "Mode",
                        "options": [
                            "SMS",
                            "Voice"
                        ],
                        "tooltip": "Mode to be used to deliver the OTP",
                        "visible": "True",
                        "editable": "True",
                        "required": "True",
                        "description": "",
                        "placeholder": ""
                    }
                ],
                "outputFields": [],
                "systemFields": []
            },
            "global": {
                "inputFields": [
                    {
                        "name": "customer_id",
                        "type": "text",
                        "title": "Customer ID",
                        "tooltip": "Customer ID for your Telesign account",
                        "visible": "True",
                        "editable": "True",
                        "required": "True",
                        "description": "",
                        "placeholder": ""
                    },
                    {
                        "name": "api_key",
                        "type": "password",
                        "title": "API KEY",
                        "tooltip": "API Key for your Telesign account",
                        "visible": "True",
                        "editable": "True",
                        "required": "True",
                        "description": "",
                        "placeholder": ""
                    },
                    {
                        "name": "sms_begin_range",
                        "type": "decimal",
                        "title": "Minimum OTP Value for SMS",
                        "tooltip": "Smallest number that can be used as an OTP while
sending a text message ",
                        "visible": "True",
                        "editable": "True",
                        "required": "True",
                        "description": "",
                        "placeholder": ""
                    },
                    {
                        "name": "sms_end_range",
                        "type": "decimal",
                        "title": "Maximum OTP Value for SMS",
```

```
                                   "tooltip": "Largest number that can be used as an OTP while
    sending a text message ",
                                   "visible": "True",
                                   "editable": "True",
                                   "required": "True",
                                   "description": "",
                                   "placeholder": ""
                               },
                               {
                                   "name": "voice_begin_range",
                                   "type": "decimal",
                                   "title": "Minimum OTP Value for voice",
                                   "tooltip": "Smallest number that can be used as an OTP while
    sending a voice message ",
                                   "visible": "True",
                                   "editable": "True",
                                   "required": "True",
                                   "description": "",
                                   "placeholder": ""
                               },
                               {
                                   "name": "voice_end_range",
                                   "type": "decimal",
                                   "title": "Maximum OTP Value for voice",
                                   "tooltip": "Largest number that can be used as an OTP while
    sending a voice message ",
                                   "visible": "True",
                                   "editable": "True",
                                   "required": "True",
                                   "description": "",
                                   "placeholder": ""
                               }
                           ],
                           "outputFields": [],
                           "systemFields": []
                       }
                   },
                   "config": {},
                   "configured": false
               },
               {
                   "nfa_id": 9,
                   "service_name": "GoogleAuthenticator",
                   "available": false,
                   "title": "Google Authenticator",
                   "traits": [
                       "trait_configure_self",
                       "trait_custom_message",
                       "trait_verify_code"
                   ],
                   "config_schema": {
                       "user": {
                           "inputFields": [
                               {
                                   "name": "email",
                                   "type": "email",
```

```
                                "title": "Email",
                                "tooltip": "",
                                "visible": "True",
                                "editable": "True",
                                "required": "True",
                                "description": "",
                                "placeholder": ""
                            }
                        ],
                        "outputFields": [
                            {
                                "name": "qr",
                                "type": "img",
                                "title": "You need Google Authenticator to complete this
  process.<br/>Scan the QR code in your app",
                                "tooltip": "",
                                "visible": "True",
                                "editable": "False",
                                "required": "True",
                                "description": "QR Code to configure Google Authenticator",
                                "placeholder": ""
                            }
                        ],
                        "systemFields": [
                            {
                                "name": "secret",
                                "type": "text",
                                "title": "Secret",
                                "tooltip": "",
                                "visible": "True",
                                "editable": "False",
                                "required": "True",
                                "description": "",
                                "placeholder": "Secret"
                            }
                        ]
                    },
                    "global": {
                        "inputFields": [
                            {
                                "name": "issuer",
                                "type": "text",
                                "title": "Issuer name",
                                "tooltip": "The issuer parameter will be used as the name of the
  provider or service this account is associated with",
                                "visible": "True",
                                "editable": "True",
                                "required": "True",
                                "description": "",
                                "placeholder": ""
                            }
                        ],
                        "outputFields": [],
                        "systemFields": []
                    }
                },
```

```
            "config": {},
            "configured": false
          }
        ]
    }
}
```

Enable or Disable an existing NFA method

Use this API to enable or disable an existing NFA method. This API is used in the "Authentication" section in the FortiSOAR UI.

Request

```
METHOD: PUT
URL: https://{HOSTNAME}/api/auth/nfamethod
BODY:
{
    "nfa_id": 13,
    "available": true,
    "code": "pune"
}
```

Response

```
"response": {}
```

Delete an existing NFA method

Use this API to delete an existing NFA method.

Request

```
METHOD: DELETE
URL: https://{HOSTNAME}/api/auth/nfamethod
BODY:
{
    "nfa_id": 16
}
```

Response

```
{
    "status": "Success",
    "data": {}
}
```

Add or Update global configurations in an existing NFA method

Use this API to add or update global configurations in an existing NFA method. This API is used in the "Authentication" section in the FortiSOAR UI.

Request

```
METHOD: POST
URL: https://{HOSTNAME}/api/auth/nfaconfig
```

```
BODY:

{
    "nfa_id": 8,
    "config": {
        "api_key": "*****************************************************************",
        "customer_id": "02CDC0C4-****-****-****-************",
        "sms_begin_range": "100",
        "sms_end_range": "999999",
        "voice_begin_range": "100",
        "voice_end_range": "99999"
    }
}
```

Response

```
{
        "status": "Success",
        "data": {
            "config": {
                "customer_id": "02CDC0C4-****-****-****-************",
                "sms_begin_range": "100",
                "sms_end_range": "999999",
                "voice_begin_range": "100",
                "voice_end_range": "99999"
            },
            "title": "Telesign",
            "service_name": "Telesign",
            "nfa_id": 8,
            "configured": true,
            "available": false
        }
    }
}
```

Get global configurations for the specified NFA method

Use this API to get the global configurations for a specific NFA method based on the given NFA ID. This API is used in the "Authentication" section in the FortiSOAR UI.

Request

```
METHOD: GET
URL: https://{HOSTNAME}/api/auth/nfaconfig
BODY:
{
    "nfa_id": 20
}
```

Response

```
{
    "config": {
        "customer_id": "02CDC0C4-****-****-****-************",
        "sms_begin_range": "100",
        "sms_end_range": "999999",
        "voice_begin_range": "100",
        "voice_end_range": "99999"
```

Get details of existing NFA users

Use this API to get details of users who are using the existing NFA method. This API is used in the "Users" section in the FortiSOAR UI.

Request

```
METHOD: GET
URL: https://{HOSTNAME}/api/auth/nfauser
```

Response

```
{
    "hydra:totalItems": 2,
    "hydra:member": [
        {
            "nfauser_id": "220",
            "nfa_id": "155",
            "uuid": "7f09a6cd-b2ea-4466-a3f6-ba6ea0de8224",
            "user_id": "215",
            "precedence": "3",
            "enabled": "False",
            "last_successful": "2023-08-01 16:23:59.134024",
            "config": {
                "phone": "919108260653",
                "mode": "sms"
            }
        },
        {
            "nfauser_id": "219",
            "nfa_id": "154",
            "uuid": "7f09a6cd-b2ea-4466-a3f6-ba6ea0de8224",
            "user_id": "215",
            "precedence": "2",
            "enabled": "True",
            "last_successful": "2023-08-01 12:00:38.091077",
            "config": {
                "qr": {
                    "mimeType": "image format",
                    "file": {
                        "data":
"iVBORw0KGgoAAAANSUhEUgAAAIIAAACCAQAAAACieC1QAAACTklEQVR4nM3WsapnOQgHYME24KsEbAO+upBW8FUE24D
j3Nmd4v699e4pv+IkMfo7B+rbo/AfC+B2P36j="
                    }
                }
            }
        }
    ]
}
```

Create a new NFA method

Use this API to create a new NFA method.

Request

```
METHOD: POST
URL: https://{HOSTNAME}/api/auth/nfamethod

 BODY:

{
     "title": "SecurityQuestion",
     "traits": [
         "trait_configure_self",
         "trait_custom_message",
         "trait_generate_code"
     ],
     "config_schema": {
         "user": {
             "inputFields": [
                 {
                     "name": "question_1",
                     "type": "text",
                     "title": "Question 1",
                     "tooltip": "",
                     "editable": "True",
                     "visible": "True",
                     "required": "True",
                     "description": "",
                     "placeholder": ""
                 },
                 {
                     "name": "question_2",
                     "type": "text",
                     "title": "Question 2",
                     "tooltip": "",
                     "editable": "True",
                     "visible": "True",
                     "required": "True",
                     "description": "",
                     "placeholder": ""
                 },
                 {
                     "name": "question_3",
                     "type": "text",
                     "title": "Question 3",
                     "tooltip": "",
                     "editable": "True",
                     "visible": "True",
                     "required": "True",
                     "description": "",
                     "placeholder": ""
                 },
                 {
                     "name": "answer_1",
                     "type": "password",
                     "title": "Answer 1",
                     "tooltip": "",
                     "editable": "True",
                     "visible": "True",
```

```
                "required": "True",
                "description": "",
                "placeholder": ""
            },
            {
                "name": "answer_2",
                "type": "password",
                "title": "Answer 2",
                "tooltip": "",
                "editable": "True",
                "visible": "True",
                "required": "True",
                "description": "",
                "placeholder": ""
            },
            {
                "name": "answer_3",
                "type": "password",
                "title": "Answer 3",
                "tooltip": "",
                "editable": "True",
                "visible": "True",
                "required": "True",
                "description": "",
                "placeholder": ""
            }
        ],
        "outputFields": [
            {
                "name": "question_1",
                "type": "text",
                "title": "Question 1",
                "tooltip": "",
                "editable": "True",
                "visible": "True",
                "required": "True",
                "description": "",
                "placeholder": ""
            },
            {
                "name": "question_2",
                "type": "text",
                "title": "Question 2",
                "tooltip": "",
                "editable": "True",
                "visible": "True",
                "required": "True",
                "description": "",
                "placeholder": ""
            },
            {
                "name": "question_3",
                "type": "text",
                "title": "Question 3",
                "tooltip": "",
                "editable": "True",
```

```
                    "visible": "True",
                    "required": "True",
                    "description": "",
                    "placeholder": ""
                }
            ],
            "systemFields": []
        },
        "global": {
            "inputFields": [],
            "outputFields": [],
            "systemFields": []
        }
    }
  }
```

Response

```
{
        "status": "Success",
        "data": {
            "nfa_id": 13,
            "service_name": "Securityquestion",
            "available": false,
            "title": "SecurityQuestion",
            "traits": [
                "trait_configure_self",
                "trait_custom_message",
                "trait_generate_code"
            ],
            "config_schema": {
                "user": {
                    "inputFields": [
                        {
                            "name": "question_1",
                            "type": "text",
                            "title": "Question 1",
                            "tooltip": "",
                            "visible": "True",
                            "editable": "True",
                            "required": "True",
                            "description": "",
                            "placeholder": ""
                        },
                        {
                            "name": "question_2",
                            "type": "text",
                            "title": "Question 2",
                            "tooltip": "",
                            "visible": "True",
                            "editable": "True",
                            "required": "True",
                            "description": "",
                            "placeholder": ""
                        },
                        {
                            "name": "question_3",
```

```
                    "type": "text",
                    "title": "Question 3",
                    "tooltip": "",
                    "visible": "True",
                    "editable": "True",
                    "required": "True",
                    "description": "",
                    "placeholder": ""
                },
                {
                    "name": "answer_1",
                    "type": "password",
                    "title": "Answer 1",
                    "tooltip": "",
                    "visible": "True",
                    "editable": "True",
                    "required": "True",
                    "description": "",
                    "placeholder": ""
                },
                {
                    "name": "answer_2",
                    "type": "password",
                    "title": "Answer 2",
                    "tooltip": "",
                    "visible": "True",
                    "editable": "True",
                    "required": "True",
                    "description": "",
                    "placeholder": ""
                },
                {
                    "name": "answer_3",
                    "type": "password",
                    "title": "Answer 3",
                    "tooltip": "",
                    "visible": "True",
                    "editable": "True",
                    "required": "True",
                    "description": "",
                    "placeholder": ""
                }
            ],
            "outputFields": [
                {
                    "name": "question_1",
                    "type": "text",
                    "title": "Question 1",
                    "tooltip": "",
                    "visible": "True",
                    "editable": "True",
                    "required": "True",
                    "description": "",
                    "placeholder": ""
                },
                {
```

```
                                "name": "question_2",
                                "type": "text",
                                "title": "Question 2",
                                "tooltip": "",
                                "visible": "True",
                                "editable": "True",
                                "required": "True",
                                "description": "",
                                "placeholder": ""
                            },
                            {
                                "name": "question_3",
                                "type": "text",
                                "title": "Question 3",
                                "tooltip": "",
                                "visible": "True",
                                "editable": "True",
                                "required": "True",
                                "description": "",
                                "placeholder": ""
                            }
                        ],
                        "systemFields": []
                    },
                    "global": {
                        "inputFields": [],
                        "outputFields": [],
                        "systemFields": []
                    }
                },
                "config": {},
                "configured": false
            }
        }
    }
```

The following image contains a sample of the request body that can be used to create a new NFA method:

```json
{
    "title": "Telesign",
    "traits": ["trait_configure_self","trait_custom_message","trait_generate_code"],
    "config_schema": {
        "global": {
            "inputFields": [ {
                    "title": "Customer ID",
                    "name": "cust_id",
                    "required": true,
                    "visible": true,
                    "editable":true,
                    "type": "text",
                    "tooltip": "Customer ID for Telesign",
                    "description": "",
                    "placeholder": ""
            }],
            "outputFields": [{…
            }],
            "systemFields": [{…
            }]
        },
        "user": {
            "inputFields": [ {…
            }],
            "outputFields": [{…
            }],
            "systemFields": [{…
            }]
        }
    }
}
```

The body of the request must contain the following keys:

- `title`: Title of the NFA method you want to create.
- `traits`: Traits that you want to assign to the NFA method: Following is the list of supported traits:
  - `trait_configure_self`: Used to add and customize user-level configuration.
  - `trait_custom_message`: Used to customize the 2FA code request message that is displayed on the login screen.
  - `trait_generate_code`: Used by the NFA method to generate a verification code.
  - `trait_verify_code`: Used to define custom verification logic as opposed to static code verification provided by the '`trait_generate_code`' trait. This trait can be used for NFA methods that are time-based.
- `config-schema`: Contains two inner dictionaries: **global** and **user**.
  The `global` dictionary is used for the global configuration of the NFA method.
  The `user` dictionary is used for user-specific configuration of the NFA method.
  These inner dictionaries have the following three keys:

- `InputFields`: Fields that users require to provide as input.
- `OutputFields`: Fields that should be displayed to users once the configuration is completed, for example, a QR code.
- `SystemFields`: Fields for internal usage, i.e., those that are not shown in UI/API calls.
  These keys are of the 'list' type and contain a list of *fields* in the following format:

```
{
    "title": "",
    "name": "",
    "required": "",
    "visible": "",
    "editable": "",
    "type": "",
    "tooltip": "",
    "description": "",
    "placeholder": ""
}
```

Complete the implementation of the generated file

Once you have added a new NFA method using `/api/auth/nfamethod`, you must complete the implementation of the generated file located at: `/opt/cyops-auth/mfa/services/<service_name>.py`. You are required to complete only the 'execute' methods for the trait classes in the implementation file. The structure of the implementation file is as follows:

All imports based on traits. This is provided by default.	```"""AUTOGENERATED PYTHON CODE"""

from enum import Enum
from mfa.traits.base.trait_base import Trait
from mfa.traits.base.trait_base import inject_trait
from mfa.traits.trait_configure_self import TraitConfigureSelf
from mfa.traits.trait_custom_message import TraitCustomMessage
from mfa.traits.trait_generate_code import TraitGenerateCode``` |
| Enums for keys in the `config- schema`. This is provided by default. The sample image displays enums for global input fields. | ```class GlobalInputConfig(Enum):
    """Enum for global config input fields"""

    # All the keys defined for global input in config schema``` |
| Classes for the traits added. This is provided by default. However, users can choose their own implementation. | The following sample image is of a class responsible for performing self-configuration: |

```
class TelesignTraitConfigureSelf(TraitConfigureSelf):
    """Implementation for trait trait_configure_self"""


    def execute(self, *args, **kwargs):
        """Implementation for trait_configure_self"""
        # TODO
        logger.info("Call received for Telesign trait_configure_self")
        user_config = kwargs.pop("user_config")
        global_config = kwargs.pop("global_config")
        return user_config
```

The following sample image is of a class responsible for customizing the 2FA message displayed on the login page:

```
class TelesignTraitCustomMessage(TraitCustomMessage):
    """Implementation for trait trait_custom_message"""


    def execute(self, *args, **kwargs):
        """Implementation for trait_custom_message"""
        # TODO
        logger.info("Call received for Telesign trait_custom_message")
        user_config = kwargs.pop("user_config")
        global_config = kwargs.pop("global_config")
        # Add your custom message here
        return {"message":"A Code has been sent to your device"}
```

The following sample image is of a class responsible for 2FA code generation:

```
class TelesignTraitGenerateCode(TraitGenerateCode):
    """Implementation for trait trait_generate_code"""


    def execute(self, *args, **kwargs):
        """Implementation for trait_generate_code"""
        # TODO
        logger.info("Call received for Telesign trait_generate_code")
        user_config = kwargs.pop("user_config")
        global_config = kwargs.pop("global_config")
        # Add your code generation logic here
        return {"code":""}
```

The following sample image is of a class responsible for running the health check for the 2FA global configuration:

```
class TelesignTraitHealthCheck(TraitHealthCheck):
    """Implementation for trait trait_health_check"""


    def execute(self, *args, **kwargs):
        """Implementation for trait_health_check"""
        # TODO
        logger.info("Call received for Telesign trait_health_check")
        global_config = kwargs.pop("global_config")
        # Add your Health Check logic here
        is_valid = True
        msg = ""
        return is_valid,msg
```

API Methods

Class for a given service. This is provided by default.

```
@inject_trait(TelesignTraitConfigureSelf)
@inject_trait(TelesignTraitCustomMessage)
@inject_trait(TelesignTraitGenerateCode)
@inject_trait(TelesignTraitHealthCheck)
class Telesign:
    """Telesign """

    pass
```

Add global configuration and activate the NFA method

Once you have completed the implementation of the generated file, you can add global configuration and activate the NFA method for FortiSOAR users. For more information, see the `Configuring Two-Factor Authentication (2FA)` topic in the *Security Management* chapter of the "Administration Guide."

For logs related to authentication issues, use the `das.log` file located at `/var/log/cyops/cyops-auth/das.log`. Additionally, if the modifications you made to the service.py file for the authentication method are not reflected, you can restart the `cyops-auth` service by using the following command:

```
# systemctl restart cyops-auth
```

FortiSOAR licensing

You can deploy the FortiSOAR enterprise license by running the `csadm` command as a *root* user: `csadm license --deploy-enterprise-license <License File Path>`. For more information on the FortiSOAR Admin CLI (`csadm`), see the *FortiSOAR Admin CLI* chapter in the "Administration Guide." For more information on licensing, see the *Licensing FortiSOAR* chapter in the "Deployment Guide." You can also deploy the FortiSOAR license using the API.

Deploying FortiSOAR license

You can deploy the FortiSOAR license using the API with or without an authorization token.

Deploying FortiSOAR license without an authorization token

From version 7.0.0 onwards, you can deploy the FortiSOAR license using the "Unauthenticate API", i.e, you can deploy the license without the need to generate the authorization token:

Request

```
POST /api/public/license HTTP/1.1
Host: fortisoar.fortinet.net
Content-Type: application/json
{
        "license_key": "<license_key>",
        "nodeId": "0961991714b6089a1d31983ea76f869e",
        "action": "deploy_license"
}
```

If the license is successfully deployed you will get the following response:

Response

```
STATUS: 200 OK
```

You can also retrieve the current status of license deployment using the following API:

Request

```
POST /api/public/license HTTP/1.1
Host: fortisoar.fortinet.net
Content-Type: application/json
{
    "nodeId": "0961991714b6089a1d31983ea76f869e",
    "action": "get_status"
}
```

Response

```
{
   "depl_status": "finished",
   "depl_info": {
      "source": "192.168.56.1",
      "user_id": null,
      "ws_session_id": null,
      "depl_start_time": "2020-12-23T12:23:30.771975"
      }
}
```

Deploying FortiSOAR license with an authorization token

Before you can deploy the license file using the API, you require to generate the authorization token. Use the following `curl` call to get the token:

```
curl -X POST \
  https://<fortisoar_host>/auth/authenticate \
  -H 'Content-Type: application/json' \
  -d '{
      "credentials":
      {
              "loginid":"<loginid>",
              "password":"<password>"
      }
}
```

To deploy the license file using the API, make the following REST call:

```
curl -X POST
  'https://<hostname>/api/auth/license
  -H 'authorization: AUTH_TOKEN'
  -H 'content-type: application/json'
  -d '{
              "license_key" : "LICENSE_KEY"
      }'
```

The `LICENSE_KEY` value is the contents of the license file generated during the license generation step. As this API is an authenticated API, it can be used only when a valid license is already deployed. A successful deployment returns an `HTTP 200 OK` status.

Retrieving the license details

From version 7.0.0 onwards, you can retrieve the important license details for a given key, even before deploying the license, using the following API:

Request

```
POST /api/public/license HTTP/1.1
Host: fortisoar.fortinet.net
Content-Type: application/json
{
  "license_key": "<license_key>",
    "action": "get_info"
}
```

Response

```
{
 "serial_no": "FSRVMPTM20000065",
 "expiry_time": "2021-05-19 00:00:00",
 "hardware_key": "4a1782123dca6394b0d97879ca98bdf0",
 "max_users": 2,

 "entitlements": {
     "branding": "advanced"
   },
 "type": "Evaluation",
 "edition": "Enterprise"
}
```

For licenses of type "Subscription", the values for `max_users` and `expiry_time` will be available only after the FortiSOAR license is deployed as the Subscription license details are fetched from the FortiGuard Distribution Network (FDN) server and therefore are available only after license deployment,

Once you have successfully deployed a FortiSOAR license, you can also retrieve the details of your license at any time using the following `curl` command:

```
curl -X GET
   'https://<fortisoar_host>/auth/license/?param=license_details'
   -H 'authorization: AUTH_TOKEN'
```

The `curl` command displays a message similar to the following:

```
{
  "expired": false,
  "expired_nodes": [],
  "remaining_days": 180,
  "details": {
    "is_distributed": false,
    "role": [
      "master"
    ],
    "entitlements": {}
  },
  "nodes": {
    "6d1a908cc06887910ed3237e0c233fb1": {
      "message": "License Details: Max Users: 10, Expiring on: 2019-11-03, Issued for 180
days",
      "details": {
```

```
          "total_users": 10,
          "remaining_users": 9,
          "expiring_on": "2019-11-03",
          "issue_time": "2019-05-07 06:26:13.368238",
          "total_days": 180,
          "remaining_days": 180,
          "customer_name": "abc",
          "is_distributed": false,
          "entitlements": {},
          "role": [
            "master"
          ]
        },
        "node": {
          "nodeId": "6d1a908cc06887910ed3237e0c233fb1",
          "nodeName": "fortisoar.localhost",
          "status": "active",
          "role": "primary",
          "currentState": "primary server"
        }
      }
    }
  }
}
```

From version 7.0.0 onwards you can also retrieve license details for a specific node using the 'node_id' parameter.
When node_id value is "SELF", then license details are returned for the node on which the API call is received.

Request

```
GET api/auth/license/?param=license_details&node_id=SELF HTTP/1.1
Host: dev.cyops.cyber
Authorization: <Bearer token>
```

Response

```
{
    "expired": false,
    "expired_nodes": [],
    "expired_nodes_details": [],
    "remaining_days": 90,
    "users": 2,
    "details": {
        "is_distributed": false,
        "role": null,
        "entitlements": {
            "branding": "advanced"
        }
    },
    "nodes": {
        "a840e36092f6edbcf9b6d97e918f81c6": {
            "details": {
                "issue_time": "2020-05-26 12:20:59.012831",
                "total_users": 2,
                "remaining_users": 1,
                "expiring_on": "2021-05-19",
                "remaining_days": 90,
                "customer_name": null,
                "is_distributed": false,
```

```
                    "entitlements": {
                        "branding": "advanced"
                    },
                    "role": null,
                    "type": "Evaluation",
                    "serial_no": "FSRVMPTM20000061",
                    "edition": "Enterprise"
                },
                "node": {
                    "nodeId": "a840e36092f6edbcf9b6d97e918f81c6",
                    "nodeName": "dev.cyops.cyber",
                    "status": "active",
                    "role": "primary",
                    "comment": "primary server",
                    "health": {}
                }
            }
        },
        "is_cloud_instance": false,
        "setup_type": "single-node"
}
```

API for administrators to forcefully log out users

This API can be used by an administrator to forcefully log out a user from FortiSOAR

```
POST /api/auth/token HTTP/1.1
Host: <host>
Content-Type: application/json
Accept: application/json
Authorization: <Bearer Token>
{
    "action": "logout",
    "users": [
        "d5d4e360-2725-40d1-9fe2-d9e9f3886a04"
    ]
}
```

Create users in bulk

Use the /api/3/insert/people API to create uses in bulk.

Request

```
METHOD: POST
URL: https://{HOSTNAME}/api/3/insert/people
BODY:
{"data":
 [
   {
     "username":"user4@gmail.com",
     "email":"user4@gmail.com",
     "firstname":"user4",
     "lastname":"4",
```

```
          "phoneMobile":"",
          "roles":[],
          "accessType":"Concurrent",
          "csActive":false,
          "loginType":6,
          "uuid":"482c47ad-edfd-42ec-b908-437cd5f86efb",
          "user":
            {
              "loginid":"user4@gmail.com",
              "user_type":6,
              "access_type":"Concurrent",
              "status":2
            }
        },
        {
          "username":"user5@gmail.com",
          "email":"user5@gmail.com",
          "firstname":"user5",
          "lastname":"5",
          "phoneMobile":"",
          "roles":[],
          "accessType":"Concurrent",
          "csActive":false,
          "loginType":6,
          "uuid":"3311f11e-2755-4076-8004-c028f17984a7",
          "user":
            {
              "loginid":"user5@gmail.com",
              "user_type":6,
              "access_type":"Concurrent",
              "status":2
            }
        }
    ]
}
```

Note: `user_type` in the `user` key specifies the type of user to be created. For example `"user_type":6` is used to creating SSO users.

View consolidated permissions for users or appliances

Use this API to view the aggregated list of effective permissions based on different roles assigned to a given user or appliance.

Request

METHOD: `GET`

URL: `https://<IP>/api/actors/permission/<actor uuid>`

Response

```
{
    "application": {
        "create": true,
        "read": true,
        "update": true,
```

```
            "delete": true,
            "execute": false,
            "fields": []
        },
        "security": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "appliances": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "people": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "routers": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "tenants": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "agents": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "contentHub": {
            "create": false,
            "read": true,
            "update": false,
```

```
                "delete": false,
                "execute": false,
                "fields": []
            },
            "workflows": {
                "create": true,
                "read": true,
                "update": true,
                "delete": true,
                "execute": true,
                "fields": []
            },
            "files": {
                "create": true,
                "read": true,
                "update": true,
                "delete": true,
                "execute": false,
                "fields": []
            },
            "activities": {
                "create": true,
                "read": true,
                "update": true,
                "delete": false,
                "execute": false,
                "fields": []
            },
            "reporting": {
                "create": true,
                "read": true,
                "update": true,
                "delete": true,
                "execute": false,
                "fields": []
            },
            "attachments": {
                "create": true,
                "read": true,
                "update": true,
                "delete": true,
                "execute": false,
                "fields": []
            },
            "comments": {
                "create": true,
                "read": true,
                "update": true,
                "delete": true,
                "execute": false,
                "fields": []
            },
            "companies": {
                "create": true,
                "read": true,
                "update": true,
```

```
            "delete": true,
            "execute": false,
            "fields": []
        },
        "fsr_queues": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "shifts": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "tasks": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "connectors": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": true,
            "fields": []
        },
        "email_templates": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "schedules": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "approvals": {
            "create": true,
            "read": true,
            "update": true,
```

```
            "delete": true,
            "execute": false,
            "fields": []
        },
        "saved_reports": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "dashboard": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "widgets": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": true,
            "fields": []
        },
        "data_archival": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "rules": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": false,
            "fields": []
        },
        "solutionpacks": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": true,
            "fields": []
        },
        "alerts": {
            "create": true,
            "read": true,
            "update": true,
```

```
        "delete": true,
        "execute": true,
        "fields": []
    },
    "announcements": {
        "create": true,
        "read": true,
        "update": true,
        "delete": true,
        "execute": true,
        "fields": []
    },
    "assets": {
        "create": true,
        "read": true,
        "update": true,
        "delete": true,
        "execute": true,
        "fields": []
    },
    "campaigns": {
        "create": true,
        "read": true,
        "update": true,
        "delete": true,
        "execute": true,
        "fields": []
    },
    "communication": {
        "create": true,
        "read": true,
        "update": true,
        "delete": true,
        "execute": true,
        "fields": []
    },
    "events": {
        "create": true,
        "read": true,
        "update": true,
        "delete": true,
        "execute": true,
        "fields": []
    },
    "hunt": {
        "create": true,
        "read": true,
        "update": true,
        "delete": true,
        "execute": true,
        "fields": []
    },
    "incidents": {
        "create": true,
        "read": true,
        "update": true,
```

```
            "delete": true,
            "execute": true,
            "fields": []
        },
        "indicators": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": true,
            "fields": []
        },
        "sla_template": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": true,
            "fields": []
        },
        "warrooms": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": true,
            "fields": []
        },
        "scenario": {
            "create": true,
            "read": true,
            "update": true,
            "delete": true,
            "execute": true,
            "fields": []
        }
    }
}
```

High Availability Health Cluster API

This API is used to retrieve information for the nodes in a High Availability (HA) cluster from the database.

`GET /api/auth/cluster/health/`: Used to retrieve health information for all the nodes in an HA cluster from the database.

`GET /api/auth/cluster/health?nodeId=<cluster-node-id>`: Used to retrieve health information for a particular node in an HA cluster from the database.

`GET /api/auth/cluster/health?latest=True&section=cpu,disk,ram,swap,all,services`: Used to retrieve the latest health information for the node who is serving the API from the database.

Model Operations

Create Individual Model Records

Request

```
METHOD: POST
URL: /api/{version}/{collection}
BODY:
{
  "field1": "value1",
  "field2": "value2"
}
```

Example

Request

```
METHOD: POST
URL: /api/3/assets
BODY:
{
  "ip": "8.8.8.8",
  "hostname": "google-public-dns-a.google.com"
}
```

Response

```
STATUS: 201
BODY:
{
  "@id": "/api/3/assets/01199609-d60f-356b-a762-129a6e1b353b",
  "@type": "Asset",
  "ip": "8.8.8.8",
  "hostname": "google-public-dns-a.google.com"
}
```

Retrieve Collection of Model Records

The responses adhere to the Hydra Spec for pagination. Records that match the specified query are returned as objects within the "hydra:member" array.

Request

```
METHOD: GET
URL: /api/{version}/{collection}
```

Response

```
STATUS: 200
BODY:
{
  "@context": "/api/{version}/contexts/{collection type}",
  "@id": "/api/{version}/{collection}",
```

```
  "@type": "hydra:PagedCollection",
  /* Contains some Hydra pagination properties */
  "hydra:member": [
    { "@id": "/api/{version}/{collection}/{uuid}", ... },
    { "@id": "/api/{version}/{collection}/{uuid}", ... },
    { "@id": "/api/{version}/{collection}/{uuid}", ... }
  ]
}
```

From version 7.0.0 onwards, the 'Pagination' response has changed as follows:

- The hydra 'next' attribute is not displayed in the API response, if there are no further results.
- The Item per page key has been removed.
- The new 'pagination' response has hydra: view with first page, next page, and last page information.

If you require the legacy view for records, though it is not recommended, you can add the $legacy_collection_ view=true query parameter in the API request. If you set the $legacy_collection_view=true parameter, then the API request will also contain the following older keys:

- 'hydra:itemsPerPage'
- 'hydra:firstPage'
- 'hydra:lastPage'

Example

Request

```
METHOD: GET
URL: /api/3/assets
```

Response

```
STATUS: 200
BODY:
{
  "@context": "/api/3/contexts/Asset",
  "@id": "/api/3/assets",
  "@type": "hydra:PagedCollection",
  "hydra:totalItems": 2,
  "hydra:itemsPerPage": 30,
  "hydra:firstPage": "/api/3/assets",
  "hydra:lastPage": "/api/3/assets",
  "hydra:member": [
    {
      "@id": "/api/3/assets/01199609-d60f-356b-a762-129a6e1b353b",
      "@type": "Asset",
      "ip": "8.8.8.8",
      "hostname": "google-public-dns-a.google.com"
    },
    {
      "@id": "/api/3/assets/017d3845-e204-367c-b74d-6bb37a8239b7",
      "@type": "Asset",
      "ip": "8.8.4.4",
      "hostname": "google-public-dns-b.google.com"
    }
```

```
    ]
  }
```

Retrieve Single Model Records

Request

```
METHOD: GET
URL: /api/{version}/{collection}/{uuid}
```

Response

```
STATUS: 200
BODY:
{
  "@id": "/api/{version}/{collection}/{uuid}",
  "@type": "{collection type}",
  "field1": "value1",
  "field2": "value2"
}
```

Example

Request

```
METHOD: GET
URL: /api/3/assets/01199609-d60f-356b-a762-129a6e1b353b
```

Response

```
STATUS: 200
BODY:
{
  "@id": "/api/3/assets/01199609-d60f-356b-a762-129a6e1b353b",
  "@type": "Asset",
  "ip": "8.8.8.8",
  "hostname": "google-public-dns-a.google.com"
}
```

Update Single Model Records

Request

```
METHOD: PUT
URL: /api/{version}/{collection}/{uuid}
BODY:
{
  "field1": "value1",
  "field2": "value2"
}
```

Example

Request

```
METHOD: GET
URL: /api/3/assets/01199609-d60f-356b-a762-129a6e1b353b
BODY:
{
  "ip": "8.8.8.8"
}
```

Response

```
STATUS: 200
BODY:
{
  "@id": "/api/3/assets/01199609-d60f-356b-a762-129a6e1b353b",
  "@type": "Asset",
  "ip": "8.8.8.8",
  "hostname": "google-public-dns-a.google.com"
}
```

Delete Single Model Records

 Deleting models can result in system instability and data model. We advise that you do not delete models after creation.

Request

```
METHOD: DELETE
URL: /api/{version}/{collection}/{uuid}
```

Example

Request

```
METHOD: DELETE
URL: /api/3/assets/01199609-d60f-356b-a762-129a6e1b353b
```

Response

```
STATUS: 204
REASON: No Content
```

Insert records in bulk

You can use the `Bulk Insert API` to insert records in bulk, in a module that you specify.

From version 7.0.0 onwards, the bulk API response has been changed to 'success' or 'failure', compared to earlier releases where multistage single, failed, and bad requests were passed.

 It is recommended that you insert up to 200 smaller records such as assets, indicators, etc. in single request and loop over your dataset. If you have larger records such as, alerts and incidents with large sourcedata, then the recommended batch size is 100.

Request

```
METHOD: POST
URL: {{YourFortiSOARHostname}}/api/{version}/insert/{moduleType}
BODY: {
            "data": [{record 1}, {record 2}]
        }
}
```

Example

Request

```
METHOD: POST
URL: {{YourFortiSOARHostname}}/api/3/insert/alerts
BODY: {
            "data": [
    {
        "name": "bulk record 1",
        "description": "<p>bulk record 1 description</p>",
        "severity": "/api/3/picklists/58d0753f-f7e4-403b-953c-b0f521eab759"
    },{
        "name": "bulk record 2",
        "description": "<p>bulk record 2 description</p>",
        "severity": "/api/3/picklists/58d0753f-f7e4-403b-953c-b0f521eab759"
    },{
        "name": "bulk record 3",
        "description": "<p>bulk record 3 description</p>",
        "severity": "/api/3/picklists/58d0753f-f7e4-403b-953c-b0f521eab759"
    }
    ]
}
```

Response: If all the records are inserted successfully, then you will get a `200 OK` response. If not all the records can be inserted, then you will get a `207` (Partially Deleted) response.

Example of a complete response:

```
{
   "@context": "/api/3/contexts/Alert",
   "@id": null,
   "@type": "hydra:Collection",
   "hydra:member": [
       {
            inserted record data
       }
   ]
}
```

Delete records in bulk

You can use the `Bulk Delete API` to delete records in bulk, in a module that you specify.

Request

```
METHOD: DELETE
URL: {{YourFortiSOARHostname}}/api/{version}/delete/{moduleType}
BODY: {
              "ids": ["value1", "value2", "value3"]
          }
}
```

Example

Request

```
METHOD: DELETE
URL: {{YourFortiSOARHostname}}/api/3/delete/alerts
BODY: {
              "ids": ["13d7115a-2d4d-4bfb-bdb8-efabfce547f011",
                              "13d7115a-2d4d-4bfb-bdb8-efabfce547f012",
                              "13d7115a-2d4d-4bfb-bdb8-efabfce547f013"]
          }
}
```

Response: If all the records are deleted successfully, then you will get a `200 OK` response. If not all the records can be deleted, if for example, you have provided an ID of an already deleted record, then you will get a `207` (Partially Deleted) response.

Update records in bulk

You can use the `Bulk Update API` to update records in bulk, in a module that you specify.

Request

```
METHOD: PUT
URL: {{YourFortiSOARHostname}}/api/{version}/update/{moduleType}
BODY: {
       "data" : [
               {
                       "field1": "value1",
                       "field2": "value2",
                       "field3": "value3"
               },
               {
                       "field1": "value1",
                       "field2": "value2",
                       "field3": "value3"
               }
       ]
}
```

Example

Request

```
METHOD: PUT
URL: {{YourFortiSOARHostname}}/api/3/update/alerts
BODY: {
```

```
        "data" : [
                {
                        "@id": "/api/3/alerts/a3abeb23-f323-45a9-856b-2bf9d10386cb",
                        "source": "1238765",
                        "description": "Test 1"
                },
                {

                        "@id": "/api/3/alerts/8c086c21-2c79-4488-ae22-d0e116469fe2",
                        "source": "097353",
                        "description": "Test 2",
                }
        ]
}
```

Response: The response will contain all the fields specified in the metadata of the module. Following is a snippet extract of a successful response:

```
STATUS: 200
BODY: {
        "data" : [
                {
                        "@id": "/api/3/alerts/a3abeb23-f323-45a9-856b-2bf9d10386cb",
                        "source": "1238765",
                        "description": "Test 1"
                },
                {

                        "@id": "/api/3/alerts/8c086c21-2c79-4488-ae22-d0e116469fe2",
                        "source": "097353",
                        "description": "Test 2",
                }
        ]
}
```

Notes:

- If you provide the body of the request appropriately, then all the records get successfully updated to the specified module, and you get a `200 OK` response.
- If you do not provide an `id` for an object (record), then the module is partially updated with the records that have an associated id. Records that do not have an id will not be updated in the specified module. In this case, you get a `207` response, which will list the records that are not updated.
- If you specify a `null` value for any field, those records get successfully updated.

Upsert records

Upsert works only if you have defined uniqueness for records in the module you want to upsert records. If you have not defined record uniqueness, the upsert operation will always insert records. Also, note that when you are upserting records, fields that are marked as non-editable also get updated and the upsert behavior for uniqueness will not work for fields that are marked as encrypted.

Single Upsert API

You can use the `Upsert API` to either update an existing record, if any record matches the unique list of fields you have specified for that module using the Module Editor, or insert a new record.

Request

Following is a sample request for a single upsert request:

```
METHOD: POST
URI: {{YourFortiSOARHostname}}/api/3/upsert/{moduleType}
BODY:
     {
             "name": "value1",
             "sourceId": null,
             "severity": "value2"
     }
```

Bulk Upsert API

You can use the `Bulk Upsert API` to either bulk update multiple records, if the records match the unique list of fields you have specified for that module, or bulk insert new records.

Request

Following is a sample request for a bulk upsert request:

```
METHOD: PUT
URI: {{YourFortiSOARHostname}}/api/3/bulkupsert/{moduleType}
BODY:[
     {
             "name": "value1",
             "sourceId": "null",
             "severity": "value2"
       },
     {
             "name": "value1",
             "sourceId": "null",
             "severity": "value2"
     }
]
```

Response: If all the records are upserted successfully, then you will get a `200 OK` response. If some records are upserted successfully and some records are not upserted, then you will get a `207` (Partial Upsert) response. If none of the records are upserted successfully, then you will get a `400` response.

Insert Feeds in Bulk

You can use the `Bulk Ingest Feed` API to insert feed records in the specified module. This API is comparatively lot faster than the previous 'Create Record' API, and you can also use it to insert records in bulk using the `createFeed` step in workflows.

Request

```
METHOD: POST
URL: {{YourFortiSOARHostname}}/api/insert-feeds/{recordType}

BODY:
{
    "data": [
        {
            "<typeOfrecord>": "<typeOfrecordValue>",
            "value": "<sample value 1>"
        },
        {
            "<typeofrecord>": "<typeOfrecordValue>",
            "value": "<sample value 2>"
        }
    ]
}
```

Example

Request

```
METHOD: POST
URL: {{YourFortiSOARHostname}}/api/ingest-feeds/indicators

BODY:
{
    "data": [
        {
            "typeofindicator": "33b6b889-4c5c-4801-898b-6c446b9568f3",
            "value": "1.1.1.3"
        },
        {
            "typeofindicator": "14b6b889-4c5c-4801-898b-6c446b9568e3",
            "value": "1.1.1.4"
        }
    ]
}
```

Response: The response contains the status of the operation with the UUID of the newly-created record. The records are internally upserted in FortiSOAR based on the unique constraint criterion defined in module editor.

💡 Using this API you cannot link the records with other records at time of recordcreation.

Example of a response:

```
{
    "status": "success",
    "uuids": [
        "dce30acb-15f9-4936-bb1c-870bc1082e97",
        "95b4a927-fb02-46d2-ae48-4d6d53b0cf0a"
    ]
}
```

In the case of existing records, the `Bulk Ingest Feed` API retains the data of existing records data and updates only the changed fields. For example, if the following data is sent:

```
"data": [
    {
       "typeofindicator": "33c6e765-4z5x-6532-564c-3j231b1234g1",
       "value": "1.1.1.5",
       "uuid": "95b4a927-fb02-46d2-ae48-4d6d53b0cf0a"
    }
```

only the `value` and `typeofindicator` fields get updated, and none of the other fields of the module get impacted.

Relationship Operations

For these examples, we will expect that an Incident has already been created in the system. All these operations will affect the `assets` field of an Incident.

Get Relationship

Request

```
METHOD: GET
URL: /api/3/incidents/bbdc13f8-015f-4615-8c63-4cebb6ec991b/assets
```

Response

```
{
  "@context": "/api/3/contexts/Asset",
  "@id": "/api/3/incidents/bbdc13f8-015f-4615-8c63-4cebb6ec991b/assets",
  "@type": "hydra:PagedCollection",
  "hydra:totalItems": 1,
  "hydra:itemsPerPage": 30,
  "hydra:firstPage": "/api/3/incidents/bbdc13f8-015f-4615-8c63-4cebb6ec991b/assets",
  "hydra:lastPage": "/api/3/incidents/bbdc13f8-015f-4615-8c63-4cebb6ec991b/assets",
  "hydra:member": [
    {
      "@id": "/api/3/assets/2ac5eafd-e7dd-465e-8ddd-6b9aeb7125c0",
      "@type": "Asset",
      "macAddress": "as:se:t2",
      "id": 639
    }
  ]
}
```

From version 7.0.0 onwards, the support for getting the count of related records by passing the `$relationshipCount` flag (`api/3/<module>/uuid/<associateModule>?Countonly=true`) on only queries has been removed. Now, to get relationships count, FortiSOAR has provided alternate route of aggregation support on `/api/query/module`, which is much faster.

Add an object to a relationship

To add an object to a relationship you need to use the `__link` key in a `PUT` request to the parent record.

Request

```
PUT /api/3/incidents/bbdc13f8-015f-4615-8c63-4cebb6ec991b
{"__link": ["/api/3/assets/5434a6e1-e9cd-4ecf-ba6b-f8ac82c3effc"]}
```

The above request links the asset object to the parent incident record.

Remove an object from a relationship

To remove an object from a relationship you need to use the `__unlink` keys in a `PUT` request to the parent record.

Request

```
PUT /api/3/incidents/bbdc13f8-015f-4615-8c63-4cebb6ec991b
{"__unlink": ["/api/3/assets/5434a6e1-e9cd-4ecf-ba6b-f8ac82c3effc"]}
```

The above request removes the removed the link of the asset object from the parent incident record.

Audit Log Purge Operations

You can use the Audit Log Purge API to purge audit logs on an automated as well as on-demand basis.

Stop Automatic Purging of Audit Logs

To stop automatic purging of audit logs, run the following API:

Request

```
Method: DELETE
URL: /api/gateway/audit/activities/ttl
Content-Type: application/json
Authorization: <Bearer Token>
```

Example:

```
DELETE /api/gateway/audit/activities/ttl
Content-Type: application/json
Authorization: <Bearer Token>
```

Response: If the API is successfully run, a `200 OK` response is returned, and the automatic purging of audit logs is stopped.

Delete audit logs before a particular date

To delete audit logs before a particular date, run the following API:

Request

```
Method: DELETE
URL: /api/gateway/audit/activities
Authorization: <Bearer Token>
Cache-Control: no-cache
```

Example:

```
DELETE /api/gateway/audit/activities?uptoDate=1524477805
Authorization: <Bearer Token>
Cache-Control: no-cache
```

Response: If the API is successfully run, a `200 OK` response is returned, and any audit logs that were present before the specified date are purged.

Export and Import Configurations Operations

From version 6.4.3 onwards, to import and export configurations, you can use the respective APIs. You can also schedule API-based configuration export and imports. Alternatively, you can also use the FortiSOAR UI to run a wizard-based export and import of configuration information. For more information on the using the wizard-based export and import, see the `Configuration Export and Import` topic in the *Application Editor* chapter of the "Administration Guide."

Importing Configurations

To import a configuration using the API, the configuration must be in the `JSON` format. Follow these steps to import a configuration:

Step 1: Create an import job.

```
POST /api/3/import_jobs
{
"status": "In Progress",
"file": "/api/3/files/<uuid of uploaded json file>"
}
```

This request creates an import job; whose output contains the `@id` of the `import_jobs` record. You can extract the UUID from the `@id`, whose format is `/api/3/import_job/<uuid>`, and use this UUID in future requests to trigger the import.

Step 2: Generate the import options.

```
GET /api/import/<import job uuid>
```

This request populates the `options` object for the import job with the default options required for import. After generating the defaults, you can modify the default options by updating the import job record with the following requests.

To get the `options` object:

```
GET /api/3/import_jobs/<import job uuid>
```

To modify the `options` object:

```
PUT /api/3/import_jobs/<import job uuid>
{
"options": (modified options object)
}
```

Step 3: Trigger the import.

```
PUT /api/import/<import job uuid>
```

This request triggers the import, which runs asynchronously. You can check the status of the import using the following request:

```
GET /api/3/import_jobs/<import job uuid>
```

Exporting Configurations

Follow these steps to export a configuration, in the `JSON` format, using the API:

Step 1: Create an export template or use an existing export template.

```
POST /api/3/export_templates
{
  "options": {
    "modules": [...],
    "viewTemplates": [...],
    "picklistNames": [...],
    "views": [...],
    "reports": [...],
    "dashboards": [...],
    "roles": [...],
    "appSettings": [...],
    "playbooks": {
      "collections": [...],
      "includeVersions": false,
      "globalVariables": [...]
    }
  },
  "name": "Configuration Export"
}
```

Following is a sample template containing data:

```
{
  "options": {
    "modules": [
      "agents"
    ],
    "viewTemplates": [
      "modules-agents-list",
      "modules-agents-detail",
      "modules-agents-form"
    ],
    "picklistNames": [
      "/api/3/picklist_names/114d88e4-ceaa-4b9f-842b-cca8db90f418",
      "/api/3/picklist_names/a7ce1d03-a4e4-4750-afdd-ca3cbb6a03b3",
      "/api/3/picklist_names/663eb9f2-d362-4640-a2f1-f5432c6d3c4e",
      "/api/3/picklist_names/a255a02f-2f4d-4535-baaf-6c64c1d6f5d0"
    ],
    "views": [
      "app"
    ],
    "reports": [
      "11ece146-396e-42f8-a2c4-358550c6f03f"
    ],
    "dashboards": [
```

```
        "2bc7fd32-1cc6-48b0-bcbb-3a7c09938d4a"
      ],
      "roles": [
        "/api/3/roles/21d7a7c6-9d68-438f-b732-e8950c5f745b"
      ],
      "appSettings": [
        "sso"
      ],
      "playbooks": {
        "collections": [],
        "includeVersions": false,
        "globalVariables": [
          "Body"
        ]
      }
    },
    "name": "Configuration Export"
}
```

Alternatively, you can also create the export template using the Configuration Export Wizard in the FortiSOAR UI and use the UUID of that template for further requests.

Step 2: Trigger the export.

`PUT /api/export?fileName=<file name ending in .json>&template=<export template uuid>`

This request triggers the export and sends the following response:

`{"@type": "ExportJob", "jobUuid": "<export job uuid>"}`

Step 3: Download the file.

`GET /api/3/export_jobs/ <export job uuid>`

To request retrieves the status of the export and sends the following response:

`{"file": {…}, "status": "Export Complete"}`

When the `status` equals "Export Complete", a file object is returned. You can download the exported file using its "File IRI".

Integration APIs

You can use 'Integration APIs' can be used in playbooks to perform various actions such as retrieving health information for connectors, creating configurations for connectors, etc.

Retrieving connector details

- To get a list of all connectors installed on a FortiSOAR instance:
  GET `api/integration/connectors/`
- To get a list of all connectors that being developed on a FortiSOAR instance:
  POST `/api/integration/connector/development/list/`
  `Body {}`
  Note: Body is Empty JSON.

- To get details of any installed connector:
  POST `api/integration/connectors/<connector_name> /<connector_version>/`
  `Body {}`
  Note: Body is Empty JSON.
  Example: `api/integration/connectors/cyops_utilities/3.1.2/`
  Note: If the specified version, 3.1.2, in our example is not installed, then the API returns details of same connector (cyops_utilities) with latest version installed, for example, version 3.1.1.

Creating configurations for any connector

To create configurations for connectors:

POST `/api/integration/configuration/`
Body
```
{
   "connector": <connector_id>,
   "connector_name": <connector_name>,
   "connector_version": <connector_version>,
   "name": <config_name>,
   "config_id": <config_id>,
   "default": <true/false>,
   "config": {
     "<connector_config_field1>": "<value>",
     "<connector_config_field2>": "<value>",
     "<connector_config_field3>": "<value>",
   },
}
```

Example of the 'Body' of a connector:

```
{
   "connector": 2,
   "connector_name": "imap",
   "connector_version": "3.5.6",
   "name": "test",
   "config_id": "b6412f73-a1b0-4446-a66a-99460133b780",
   "default": true,
   "config": {
     "host": "imap.gmail.com",
     "port": "993",
     "username": "demo@gmail.com",
     "password": "test@123",
     "ssl": true,
     "source": "INBOX",
     "destination": "PROCESSED",
     "verify": true,
     "notification_service": ""
   }
}
```

Notes:

- You must specify either the `connector_id` or `connector_name`.
- You must specify the `connector_version`.
  If `connector_version` specified is invalid, then the latest installed version of same connector is considered.
- It is not mandatory to provide the `config_id`, as a UUID gets automatically generated.

Retrieving health check details for a connector

To get the heath check details for a connector:
GET `/api/integration/connectors/healthcheck/<connector_name>/<connector_version>/?config=<config_id>`

Example: `/api/integration/connectors/healthcheck/imap/3.5.6/?config=a7c7df29-1f5c-426e-981b-b1ab1b7692cb`

Note: If you specify an invalid `config_id`, then the health check details of the 'default' configuration of the specified connector is returned.

Deleting a configuration of a connector

To delete a specific configuration of a connector:
DELETE `/api/integration/configuration/<id>/`

Example: `/api/integration/configuration/8`

Executing an action of a installed connector

To execute an action of an installed connector:

```
POST /api/integration/execute/
Body
{
    "connector": <connector_id>,
    "version": <connector_version>,
    "config": <config_id>,
    "operation": "<action_name>",
    "params": {
      "<action_param1>": <value>,
      "<action_param2>": <value>
    },
}
```

Example of the 'Body' of a connector:

```
{
    "connector": "imap",
    "connector_version": "3.5.6",
    "config": "a7c7df29-1f5c-426e-981b-b1ab1b7692cb",
    "operation": "fetch_email_new",
    "params": {
      "limit_count": 30,
      "parse_inline_image": true
    }
}
```

Uninstalling a connector

To uninstall a connector:

DELETE `/api/integration/connectors/<id>/`

Example: `/api/integration/connectors/15`

Workflow Execution

You can use the Workflow Execution API to debug playbook execution, apply filters on running playbooks, retrieve playbook IDs, list execution logs, etc.

Workflow Execution History

`GET /api/wf/api/workflows/?format=json`: Used to list the workflow execution history.

`GET /api/wf/api/workflows/29/?format=json`: Used to retrieve the detailed view of a particular workflow execution history based on the ID specified.

You can filter the workflow execution history by specifying various query parameters such as, ordering, and filters such as status. An example of the Workflow Execution API with parameters and filters is: `GET /api/wf/api/workflows/?format=json&status=finished&ordering=-created`.

Following query parameters are supported:

- **Offset** - The initial index from which to return the results.
- **Limit** - The maximum number of results to return per page.
- **Ordering** - The field using which you want to sort the results. For example, `ordering=-created` sorts the results in the descending order based on the creation date.

Following filters are supported:

- **template_iri** – Filters the workflow execution history based on the playbook IRI specified, i.e., retrieves the execution history for a particular playbook. For example, `/api/wf/api/workflows/?template_iri= "/api/3/workflows/"`.
- **records** - Filters the workflow execution history based on the record ID specified, i.e., retrieves the execution history for playbooks execution on a specific record. For example, `/api/wf/api/workflows/?records= "/api/3/alerts/alert_id"`.
- **status** – Filters the workflow execution history based on the running status of the playbooks. The running state of the playbooks can be incipient, active, failed, finished, awaiting, skipped, or terminated. For example, `/api/wf/api/workflows/?status=failed`, retrieves the execution history of all playbooks that have failed.
- **?created_after=""&created_before=""** – Filters the workflow execution history based on the time range specified. For example, `/api/wf/api/workflows/?created_after=2020-11-08&created_before=2020-11-10`, retrieves the execution history of all playbooks that are executed between the 8th and 10th of November, 2020.
- **?parent__isnull=True** – Filters the workflow execution history to retrieve the history of only the parent playbooks, i.e., specifying this filter will filter out the execution history of all children playbooks.
- **tags_exclude="",&tags_include=""** – Filters the workflow execution history based on the comma-separated list of tags specified. You can specify that you want to include the execution history of playbooks containing particular tags or you want to exclude the execution history of playbooks containing particular tags or a combination of both. For example, `/api/wf/api/workflows/?tags_exclude=system,&tags_include=ingestion,critical`

retrieves the execution history of all playbooks that contain the ingestion or critical tags and excludes all playbooks containing the system tag.

Workflow Execution Log

FortiSOAR version 6.4.4 has enhanced the Workflow Execution Log API (`/api/wf/api/query/workflow_logs/`) to fetch data from the database based on user configuration. The `workflow_logs` API now supports queries in the POST request. If `{'query': {}}` is found in the POST body of the API call, then the query set is formed based on the users' input data. Otherwise, the query works as per the current functionality, i.e., the query picks up parameters from the URL.

An example of a query containing the supported filters follows:

```
{
    "logic": "AND/OR",
    "limit": "30",
    "sort": [{
        field: "modified/status/name"
        direction: "asc/desc"
    }],
    "filters": [{
        "field": "status/user/tags/modified/name",
        "operator": "eq/contains/neq/ncontains/gte/lte",
        "value": ""
    }],
    "aggregates": [{
            "operator": "groupBy/count",
            "field": "status/user/name/id"
        }
    ]
}
```

An example of a query containing nested filters follows:

```
 "logic": "OR",
    "limit": 30,
    "filters": [{
        "logic": "OR",
        "filters": [{
            "field": "tags",
            "operator": "contains",
            "value": "test"
        }, {
            "field": "tags",
            "operator": "contains",
            "value": "testing"
        }]
    }, {
        "logic": "AND",
        "filters": [{
            "field": "status",
            "operator": "eq",
            "value": "finished"
        }, {
            "field": "status",
            "operator": "eq",
```

```
                "value": "failed"
        }]
    }]
}
```

Deleting workflow logs and cleaning up the workflow table

You can also use the `Workflows` API to retrieve details of an existing schedule used to purge workflow logs, update details and configuration of an existing schedule used to purge workflow logs, and also delete workflow logs.

You can use the '`Workflow Table Cleanup`' API (new in version 7.0.2) to shrink the space of the workflow space and make unused space available to system.

Deleting workflow logs

You can use the `Workflows` API to delete workflow logs along with a date filter.

Request

```
METHOD: DELETE

URL: /api/wf/api/workflows/delete/?format=json
BODY:
{
  "date": "<Before_Date_filter>",
  "userId": "<user_id>",
  "user": "<user_name>"
  "exclude_awaiting_wf": <true/false>
}
```

Response

```
{"result": "deleting workflows"}
```

The following filters are supported:

- `date`: The date before which the workflow logs should be deleted.
- `exclude_awaiting_wf`: Whether or not workflow logs that are in the '`awating`' state should be excluded from deletion. If you specify true then the workflow logs that are in the 'awating' state are excluded from deletion, i.e., they are not deleted.

Example

```
Delete /api/wf/api/workflows/delete/?format=json
Body
{

  "date": "2021-05-03T18:30:00.000Z",
  "userId": "3451141c-bac6-467c-8d72-85e0fab569ce",
  "user": "CS Admin",
  "exclude_awaiting_wf": true
}
```

Cleanup the Workflow table

The 'Workflow Table Cleanup' API clears all the empty rows in tables and indexes and makes the unused space available to the system, which helps in improving the performance by optimizing the space.

POST /api/wf/api/workflows/auto_vacuum/?format=json: Used to clean up the workflow table and all the tables that are related to the workflow table:

Request

```
METHOD: POST
URL: /api/wf/api/workflows/auto_vacuum/?format=json
BODY:
{
  "date": "2021-07-22 09:00:35",
  "userId": "<current_users_id>",
  "user": "<current_user_name>"
}
```

Log Collection Operations

There are two APIs introduced in version 7.0.0 for collecting FortiSOAR logs. Log collection can be a long-running task, so the API should be asynchronous and should not wait for the log collection operation to be completed. Therefore, two APIs are added: One API starts the process of collecting logs and the other one downloads the collected log file.

The logs get collected in the /opt/cyops-api/workspace/logs directory and users require Security Read permission to perform the log collection operations.

Triggering log collection

To start the process of log collection, use the following request:

Request

```
POST/api/3/utils/log-collect HTTP/1.1
Host: <account_id>.fortisoar.forticloud.com
Authorization: <Bearer Token>
{
    "password": "Test"
}
```

Note: The Password parameter is optional, i.e., you do not need to input the password.

Response

```
{
    "filename": "fortisoar-logs.tar.gz.gpg"
}
```

The filename is "fortisoar-logs.tar.gz.gpg" if you have supplied a password in the request. Else, the filename is "fortisoar-logs.tar.gz".

If the log collection is triggered successfully, then status code `200` is returned. If another log collection request is running, then status code `202` is returned.

Downloading the collected log file

To download the collected log file, use the following request:

Request

```
GET /api/3/utils/log-collect?filename=<filename> HTTP/1.1
Host: <account_id>.fortisoar.forticloud.com
Authorization: <Bearer Token>
```

Response

A file is downloaded with the name `fortisoar-logs.tar.gz.gpg` (or `fortisoar-logs.tar.gz`) according to the input provided (password specified or not) in the API request, if corresponding previous log collection operation was successful.

If log collection process is still running, then status code `202` is returned. If the log collection file is not found, then status code `400` is returned.

Notification Framework APIs

Channel APIs

Get List of all Channels

Use this API to retrieve a list of all notification channels in your system. The API response contains the list of channels in the `hydra:member` property.

Request

METHOD: `GET`

URL: `/api/rule/api/channel/`

Response

```
{
  "@context": "/rule/api/channel/context/",
  "hydra:nextPage": null,
  "hydra:previousPage": null,
  "hydra:totalItems": 6,
  "hydra:itemsPerPage": 10,
  "hydra:firstPage": "/rule/api/channel/",
  "hydra:lastPage": "/rule/api/channel/?offset=0",
  "hydra:member": []
}
```

Update Channel

Use this API to update any non-system channel, i.e., any channel created by a user. The API will return the updated channel in its response.

Request

METHOD: PUT

URL: /api/rule/api/channel/<channel_uuid>/

BODY:

```
{
        "type": "connector",
        "name": "demo channel",
        "description": "This is the demo channel",
        "config": {
                "config": "88c3d39c-2fa9-4731-b00d-29815008f17c",
                "version": "2.4.3",
                "label": "SMTP",
                "connector": "smtp",
                "operation": "send_email"
        },
        "retryCount": 0,
        "is_active": true,
        "default_params": {
                "to_recipients": "",
                "cc_recipients": "",
                "bcc_recipients": "",
                "body": "",
                "subject": "",
                "iri_list": ""
        }
}
```

Create Channel

Use this API to create a custom channel that contains the configuration for any rule that requires to be executed.

Following is an example to create a 'connector' type channel that contains the configuration details of the connector that is used in the channel.

Request

METHOD: POST

URL: /api/rule/api/channel/

BODY:

```
{
        "type": "connector",
        "name": "demo channel",
        "description": "This is the demo channel for updating",
        "config": {
                "config": "88c3d39c-2fa9-4731-b00d-29815008f17c",
                "version": "2.4.3",
```

```
              "label": "SMTP",
              "connector": "smtp",
              "operation": "send_email"
      },
      "retryCount": 0,
      "is_active": true,
      "default_params": {
              "to_recipients": "demo@fortinet.com",
              "cc_recipients": "",
              "bcc_recipients": "",
              "body": "",
              "subject": "",
              "iri_list": ""
      }
}
```

Delete Channel

Use this API to delete any non-system channel, i.e., any channel created by a user.

Request

METHOD: DELETE

URL: /api/rule/api/channel/<channel_uuid>/

BODY: {}

Rules APIs

Get List of all Rules

Use this API to retrieve a list of all rules in your system. The API response contains the list of rules in the hydra:member property.

Request

METHOD: GET

URL: /api/rule/api/rules/

Response

```
{
  "@context": "/rule/api/rules/context/",
  "hydra:nextPage": null,
  "hydra:previousPage": null,
  "hydra:totalItems": 6,
  "hydra:itemsPerPage": 10,
  "hydra:firstPage": "/rule/api/rules/",
  "hydra:lastPage": "/rule/api/rules/?offset=0",
  "hydra:member": []
}
```

Create Rule

Use this API to create a custom rule. The API will return the created rule in its response.

Request

METHOD: POST

URL: /api/rule/api/rules

BODY:

```
{
        "name": "demo rule",
        "entity_type": "alerts",
        "event_type": "create",
        "trigger_condition": {
                "sort": [],
                "limit": 30,
                "logic": "AND",
                "filters": [{
                        "field": "status",
                        "operator": "eq",
                        "value": "/api/3/picklists/fac53e73-8d16-4189-98d5-95fbd1555232",
                        "_value": {
                                "itemValue": "Closed",
                                "@id": "/api/3/picklists/fac53e73-8d16-4189-98d5-95fbd1555232"
                        },
                        "type": "object"
                }]
        },
        "actions": [{
                "type": "connector",
                "params": {
                        "body": "",
                        "subject": "",
                        "iri_list": "",
                        "cc_recipients": "",
                        "to_recipients": "demo@fortinet.com",
                        "bcc_recipients": ""
                },
                "channel_uuid": "8b8d5061-4cd2-4366-9f5a-6db47c7a18a6",
                "enabled": true
        }],
        "is_system": false,
        "is_active": true,
        "priority": 10,
        "event_source": "crudhub",
        "source": null,
        "channel_preference_field": null,
        "visible": true,
        "category": "generic",
        "expiry": null,
        "entity_id": null,
        "parent_rule": null,
        "workflow": null,
        "entity_name": "Alerts"
}
```

Update Rule

Use this API to update any rule. The API will return the updated rule in its response.

Request

METHOD: PUT

URL: /api/rule/api/rules/<rule_uuid>/

BODY: Same as the body of the 'Create Rule' API but with updated fields.

Delete Rule

Use this API to delete any rule.

Request

METHOD: DELETE

URL: /api/rule/api/rules/<rule_uuid>/

BODY: {}

Bulk Rule Delete

Use this API to delete multiple rules at once by specifying a list containing the UUIDs of the rules you want to delete.

Request

METHOD: DELETE

URL: /api/rule/api/rules/bulk/delete

BODY: {"uuids":<list_of_rule_uuids>}

Response:
{"message":"Rules are been deleted Successfully"}

Clone Rules

Use this API to clone multiple rules at once by specifying a list containing the UUIDs of the rules you want to clone.

Request

METHOD: POST

URL: /api/rule/api/rules/bulk/clone

BODY: {"uuids":[<list_of_rule_uuids>]}

Response:
{"message":"Rules are been cloned Successfully"}

Bulk Update Rules

Use this API to update multiple rules at once by specifying a list containing the UUIDs of the rules you want to update and the parameters you want to update.

Request

METHOD: `PUT`

URL: `/api/rule/api/rules/bulk/update`

BODY:

```
{
      "uuids": [<list_of_rule_uuids>],
      "payload": {
            "is_active": false
      }
}
```

Response:
```
{"message":"Rules are been updated Successfully"}
```

Get all the system notifications

Use this API to retrieve a list of all notifications from your system. The API will return the list of all system notifications in its response.

Request

METHOD: `POST`

URL: `/api/rule/api/system-notification/notifications/?format=json`

BODY: `{}`

Supported Query parameters

- `read`
  The `read` parameter takes boolean values 'true' or 'false' and returns all the system notification that are not-read or read.
  For example the following API call:
  `/api/rule/api/system-notification/notifications/?format=json&read=false`
  returns all unread notifications

- `entity_type__in`
  The `entity_type__in` parameter takes either a single string or a comma-separated string of multiple values and returns all the system notification belongs to the given entity type.
  For example the following API call:
  `/api/rule/api/system-notification/notifications/?format=json&entity_type__in=approvals,manualinput`
  returns all notifications of type 'approval' or 'manual input'.

Mark system notification as read

Use this API to mark a specific system notification as read on your FortiSOAR system by specifying the UUID of the system notification you want to mark as read.

Request

METHOD: POST

URL: /api/rule/api/system-notification/read/?format=json

BODY: {"uuid": <uuid_of_system_notification>}

Purging Notifications APIs

Purging of System Notification

Use this API to purge system notifications from your FortiSOAR system.

Purge all read system notifications

Use this API to delete all system notifications that are read from your FortiSOAR system.

Request

METHOD: POST

URL: /api/rule/api/system-notification/purge/?format=json&read=true

BODY: {}

Response:
{"result": "System Notification purge started", "status": "started"}

Purge all system notifications

Use this API to delete all system notifications, whether read or unread, from your FortiSOAR system.

Request

METHOD: POST

URL: /api/rule/api/system-notification/purge/?format=json

BODY: {}

Response:
{"result": "System Notification purge started", "status": "started"}

Purging of Notification Failure logs

Use this API to delete notification failure logs from your FortiSOAR system that are older than the date specified in the payload.

Request

METHOD: `DELETE`

URL: `/api/rule/api/pending-notification/purge`

BODY: `{"date":"2022-06-20T07:58:01.447Z","userId":"<user_uuid>","user":"<user_name>"}`

Response:
`{"result": "Failed Notification Logs purging initiated", "status": "started"}`

Schedule API

Triggering a schedule immediately

You can trigger a schedule immediately using the '`scheduled`' API.

Request

```
METHOD: POST
URL: /api/wf/api/scheduled/trigger-now/
```

Response

If the schedule is successfully triggered then the API will give the following response:

`{"message":"The associated schedule is successfully triggered"}`

Query API Reference

API Routes

Apply Basic Query

```
GET /api/3/{collection}?param1=value&param2=value
```

For more information, see *API Endpoints Reference* chapter.

Apply Ad Hoc Query

```
POST /api/query/{collection}
BODY:
{ QUERY OBJECT }
```

- The `collection` parameter matches the collection route in the typical CRUD API (/api/3/{collection})

Example

```
POST /api/query/incidents
BODY:
{
  "logic": "AND",
  "filters": [
    {
      "field": "status.itemValue",
      "operator": "eq",
      "value": "Open"
    }
  ]
}
```

Apply Persisted Query

```
GET /api/query/{collection}/{queryId}
```

- The `collection` parameter matches the collection route in the typical CRUD API (`/api/3/{collection}`)
- The `queryId` parameter matches a Query Object UUID stored in `/api/3/query_objects`

Example

```
GET /api/query/incidents/2e77a714-f0c1-45ca-bd49-b71efbd9328c
```

Query Objects

Filter

A filter contains a field, operator, and a value.

```
{
"field": "{fieldName}",
"operator": "{operator}",
"value": {value}
}
```

Field

The field can be any valid field in the module. To access sub-elements of a relationship or picklist field, you can use dot notation or double-underscore notation.

```
"field": "name"
"field": "status.itemValue"
"field": "assignedToPerson.email"
```

Operator

The operator can be any of the available operators for a specific field type.

```
"operator": "eq"
"operator": "like"
```

For the complete list of supported operations, see the Supported Field Operators section.

Value

The value is the object of the filter or comparison. For example, if you are using the `is like` operator, then the value will be a pattern to match the record field.

```
"operator": "eq",
"value": "Alert 123: Repeated login failures - device xxx"

"operator": "like",
"value": "alert ___: Repeated login failures%"
```

For the complete list of supported operations, see the Supported Field Operators section.

Supported Field Operators

Operator	Value Syntax	Description

`eq`	string	**Equals**: Field is an exact match.
`neq`	string	**Not Equals**: Field is not an exact match.
`lt`	number	**Less Than**: Field has a lesser than value. In the case of `Date/Time` fields this translates to "Before", i.e. the date field is before the specified date.
`lte`	number	**Less Than or Equal To**: Field has a lesser than or equal to value. In the case of `Date/Time` fields this translates to "On or Before", i.e. the date field is on or after the specified date.
`gt`	number	**Greater Than**: Field has a greater than value. In the case of `Date/Time` fields this translates to "After", i.e. the date field is after the specified date.
`gte`	number	**Greater Than or Equal To**: Field has a greater than or equal to value. In the case of `Date/Time` fields this translates to "On or After", i.e. the date field is on the specified date or after the specified date.
`in`	value1 | value2 ...	**In**: Field is in the given list of values.
`nin`	value1 | value2 ...	**Not In**: Field is not in the given list of values.
`contains`	string	**Contains**: Object contains the key. Only applicable for field type *Object*.
`like`	string pattern	**Like**: Field matches the given pattern. The pattern can contain text, percent (`%`), or an underscrore (`_`) `%` represents zero or more of any characters. `_` represents one character.
`notlike`	string pattern	**Not Like**: Field does not match the given pattern. The pattern can contain text, percent (`%`), or an underscrore (`_`) `%` represents zero or more of any characters. `_` represents one character.
`isnull`	boolean	**Is Null**: If value is set as true, then this checks if the field is null. If value is set as false, then this checks if the field is not null.

Logic

AND

A collection of filters can be applied in conjunction using `"logic": "AND"`

Example

Query all records where `record.assignedToUser !== null && record.status === "Open"`

```
{
  "logic": "AND",
  "filters": [
    {
      "field": "assignedToUser",
      "operator": "isnull",
```

```
      "value": false
    },
    {
      "field": "status",
      "operator": "eq",
      "value": "Open"
    }
  ]
}
```

OR

A collection of filters can be applied in disjunction using `"logic": "OR"`

Example

Query all records where `record.status === "Open" || record.status === "Pending"`

```
{
  "logic": "OR",
  "filters": [
    {
      "field": "status",
      "operator": "eq",
      "value": "Open"
    },
    {
      "field": "status",
      "operator": "eq",
      "value": "Pending"
    }
  ]
}
```

Nested Logic

Nesting logic filters allows you to build queries with nested logic like `conditionX && (conditionY || conditionZ)`

Example

Query all records where `record.assignedToUser !== null && (record.status === "Open" || record.status === "Pending")`

```
{
  "logic": "AND",
  "filters": [
    {
      "field": "assignedToUser",
      "operator": "isnull",
      "value": false
    },
    {
      "logic": "OR",
      "filters": [
```

```
      {
        "field": "status",
        "operator": "eq",
        "value": "Open"
      },
      {
        "field": "status",
        "operator": "eq",
        "value": "Pending"
      }
    ]
  }
 ]
}
```

Sort

Records can be sorted by field/direction using the `"sort": []` list. This list contains objects with `"field"` and `"direction"` keys. The `"field"` key identifies which field (or association) to sort by. The `"direction"` key identifies whether to sort ascending (`"ASC"`) or descending (`"DESC"`). Order is maintained when applying these sorts.

```
{
  "logic": "AND",
  "filters": [],
  "sort": [
    {
      "field": "createDate",
      "direction": "DESC"
    }
  ]
}
```

Aggregation

Records can be aggregated using the `"aggregates": []` list. This list contains objects with `"operator"`, `"field"`, and `"alias"` keys. The `"operator"` key identifies which aggregate operator to apply to the specified field. The `"field"` key identifies the field to apply it to. The `"alias"` key identifies what the returned field looks like.

Supported Aggregate Operators

- `fields`
- `select`
- `count`
- `countdistinct`
- `groupby`
- `distinct`
- `sum`
- `max`
- `min`

- `median`
- `avg`

Examples

Count all records in each status grouping

```
{
  "logic": "AND",
  "filters": [],
  "aggregates": [
    {
      "operator": "groupby",
      "field": "status",
      "alias": "status"
    },
    {
      "operator": "countdistinct",
      "field": "*",
      "alias": "total"
    }
  ]
}
```

Average 'resolve' time between an incident's 'closed' and 'resolved' date

The following sample API gives you the **'Average'** for the 'resolve' time between an incident's closed and resolved date. You can optionally add additional filters if required.

Request

URL: `https://<Host>/api/query/incidents?$limit=30`

BODY:

```
{
    "sort": [],
    "limit": 30,
    "logic": "AND",
    "filters": [
        {
            "logic": "AND",
            "filters": [
                {
                    "field": "resolveddate",
                    "operator": "gte",
                    "type": "primitive",
                    "value": "2022-04-22T09:39:39.358Z"
                },
                {
                    "field": "resolveddate",
                    "operator": "lte",
                    "type": "primitive",
                    "value": "2022-05-22T09:39:39.358Z"
                }
            ],
```

```
                    "type": "datetime"
                }
            ],
            "aggregates": [
                {
                    "operator": "avg",
                    "field": "resolveddate,createDate",
                    "alias": "value"
                }
            ]
    }
```

Response:

```
{
    "@context": "/api/3/contexts/Incident",
    "@id": "/api/3/incidents",
    "@type": "hydra:Collection",
    "hydra:member": [
        {
            "value": "00:00:37.071302"
        }
    ],
    "hydra:totalItems": 1,
    "hydra:view": {
        "@id": "/api/query/incidents?%24limit=30",
        "@type": "hydra:PartialCollectionView"
    }
}
```

Total time taken between when an incident is created and when it is resolved

The following sample API gives you the '**Sum**' of an incident's 'create' and 'resolve' date.

Request

URL: `https://<Host>/api/query/incidents?$limit=30`

BODY:

```
{
    "sort": [],
    "limit": 30,
    "logic": "AND",
    "aggregates": [
        {
            "operator": "sum",
            "field": "resolveddate,createDate",
            "alias": "value"
        }
    ]
}
```

Response:

```
{
    "@context": "/api/3/contexts/Incident",
    "@id": "/api/3/incidents",
    "@type": "hydra:Collection",
```

```
    "hydra:member": [
        {
            "value": "00:01:51.213906"
        }
    ],
    "hydra:totalItems": 1,
    "hydra:view": {
        "@id": "/api/query/incidents?%24limit=30",
        "@type": "hydra:PartialCollectionView"
    }
}
```

Minimum time taken between when an incident is created and when it is resolved

The following sample API gives you the '**Min**' of an incident's 'create' and 'resolve' date.

Request

URL: `https://<Host>/api/query/incidents?$limit=30`

BODY:

```
{
    "sort": [],
    "limit": 30,
    "logic": "AND",
    "aggregates": [
        {
            "operator": "min",
            "field": "resolveddate,createDate",
            "alias": "value"
        }
    ]
}
```

Response

```
{
    "@context": "/api/3/contexts/Incident",
    "@id": "/api/3/incidents",
    "@type": "hydra:Collection",
    "hydra:member": [
        {
            "value": "00:01:51.213906"
        }
    ],
    "hydra:totalItems": 1,
    "hydra:view": {
        "@id": "/api/query/incidents?%24limit=30",
        "@type": "hydra:PartialCollectionView"
    }
}
```

Maximum time taken between when an incident is created and when it is resolved

The following sample API gives you the '**Max**' of an incident's 'create' and 'resolve' date.

Request

URL: `https://<Host>/api/query/incidents?$limit=30`

BODY:

```
{
    "sort": [],
    "limit": 30,
    "logic": "AND",
    "aggregates": [
        {
            "operator": "max",
            "field": "resolveddate,createDate",
            "alias": "value"
        }
    ]
}
```

Response:

```
{
    "@context": "/api/3/contexts/Incident",
    "@id": "/api/3/incidents",
    "@type": "hydra:Collection",
    "hydra:member": [
        {
            "value": "00:01:51.213906"
        }
    ],
    "hydra:totalItems": 1,
    "hydra:view": {
        "@id": "/api/query/incidents?%24limit=30",
        "@type": "hydra:PartialCollectionView"
    }
}
```

Median time taken between when an incident is created and when it is resolved

The following sample API gives you the '**Median**' of an incident's 'create' and 'resolve' date.

Request

URL: `https://<Host>/api/query/incidents?$limit=30`

BODY:

```
{
    "sort": [],
    "limit": 30,
    "logic": "AND",
    "aggregates": [
        {
            "operator": "median",
            "field": "resolveddate,createDate",
            "alias": "value"
        }
    ]
}
```

Response:

```
{
    "@context": "/api/3/contexts/Incident",
    "@id": "/api/3/incidents",
    "@type": "hydra:Collection",
    "hydra:member": [
        {
            "value": "00:01:51.213906"
        }
    ],
    "hydra:totalItems": 1,
    "hydra:view": {
        "@id": "/api/query/incidents?%24limit=30",
        "@type": "hydra:PartialCollectionView"
    }
}
```

Associations

Wherever a `"field"` key is specified in a query object, you can use dot-notation to query against an associated entity using the same operators that are available for the root record.

Example

Query all records assigned to a user named Jeff.

```
{
  "logic": "AND",
  "filters": [
    {
      "field": "assignedToUser.firstname",
      "operator": "eq",
      "value": "Jeff"
    }
  ]
}
```

Model Type

Use this API to search for records from different models like Incident, Alerts or any other custom models based on multiple valid fields in the model.

The API is: `https://{{hostname}}/api/query/{{modelType}}`. For example, `https://{{hostname}}/api/query/incidents` where incidents is a model in which you want to search for records. This API uses POST as the request method.

Example

```
BODY:
{
    "aggregates": [
        {
            "operator": "groupby",
            "field": "phase.itemValue",
            "alias": "phase"
        },
```

```
      {
            "operator": "avg",
            "field": "dwellTime",
            "alias": "maxDwellTime"
      }
    ],
    "logic": "AND",
    "filters": []
}
```

Response: JSON list of records matching the filter criteria provided in the `BODY` of the `modelType`.

Get only specified fields in the API response body while querying a module

Use the "`__selectFields`" parameter in the request body of '`query`' APIs such as `/api/query/alerts`, to get only the fields you have specified in the API response body when you exporting or fetching alert records using the API.

Example

`__selectFields: ["id", "severity"]`

Response: The API response body will contain only the "id" and "severity" fields when you exporting or fetching alert records using the API.

Ignore specified fields from the API response body while querying a module

Use the "`__ignoreFields`" parameter in the request body of '`query`' APIs such as `/api/query/alerts`, to ignore the fields you have specified from the API response body when you exporting or fetching alert records using the API.

Example

`__ignoreFields: ["createDate", "createUser"]`

Response: The API response body of the exported alert records will contain all fields except the "createDate" and "createUser" fields.