TRACKPLANFM API

Integration guide

Version	Date	Author	Description of Changes
1.0	17/04/2025	TrackplanFm	Initial document creation

Contents

Overview

The TrackplanFM API enables developers to programmatically interact with the TrackplanFM CAFM system. This document provides a general overview of the API's base URL, available endpoints, supported authentication methods, and request/response formats to facilitate seamless integration with the platform.

This documentation should be treated as a general reference only. For the most up-to-date and comprehensive information, please refer to:

- The official OpenAPI documentation available via Swagger at:

  https://{Server Name}/api/swagger

- This Swagger interface provides interactive documentation, request/response examples, and allows you to test API endpoints directly

Developers should always refer to Swagger documentation as the authoritative source when implementing integrations with the TrackplanFM API.

Base URL

https://{Server Name}

Note: To browse to swagger, append /api/swagger to the end of the base URL

1 Authentication

The TrackplanFM API supports two methods of authentication, catering to both user-based and service-based access:

1.1 JWT Authentication

JWT (JSON Web Token) authentication is used when access is tied to a specific user account in the TrackplanFM system. This method provides secure, time-limited tokens that inherit the permissions of the authenticated user.

Key points about JWT authentication:

Each token represents a specific user's session and permissions - Tokens expire after a set period for security - Appropriate for user-facing applications or when actions need to be tracked to specific users

Implementation:

1. Send a POST request to /api/auth/login with user credentials

2. Store the returned JWT token

3. Include the token in all subsequent API requests using the header:

Authorization: Bearer {your_jwt_token}

This method is ideal for applications where access is tied to individual user roles and permissions.

1.2 API Key Authentication

API Key authentication is intended for service-level access and does not require user login for each request. API keys are typically used for server-to-server communication or automated processes.

Requirements:

The client must have API access and generate an API key. Contact your Trackplan Administrator to have this enabled if not visible on your site.

Once appropriate permissions are granted. The API key can be accessed from the API Key panel in the Settings area of the TrackplanFM site.

Figure 1.1 : Access Panel in general settings for Api

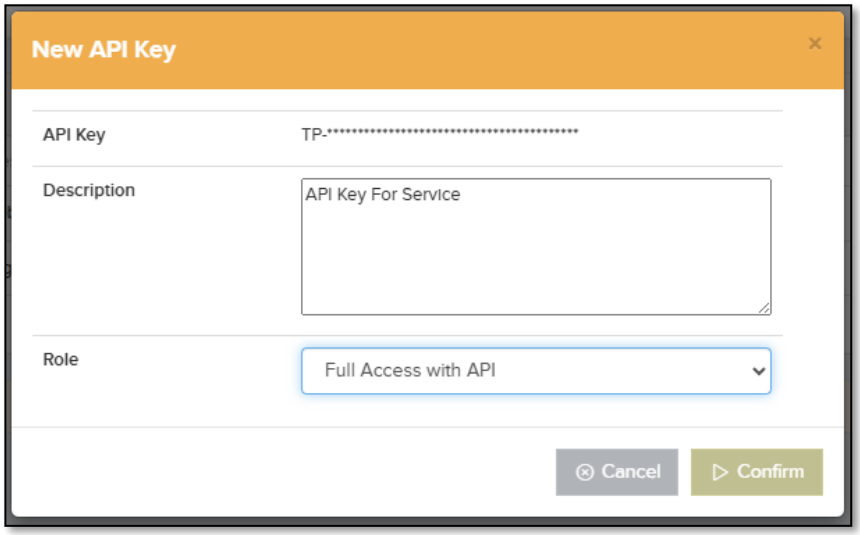

Figure 1-1:Generating Api Key for Client Usage

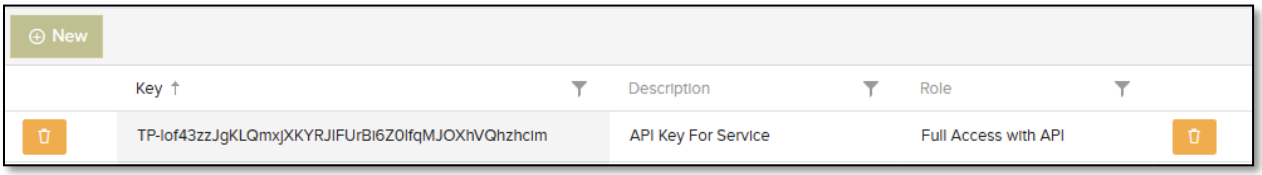

Figure 1-2: Generated Api Key for Client

Once generated copy the key securely and paste in the swagger authentication form.

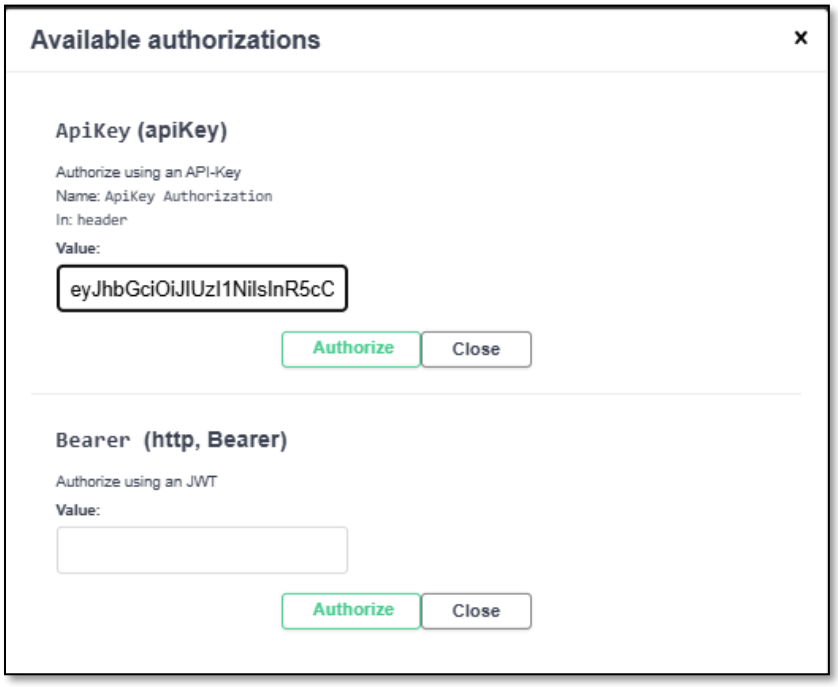

Figure 1-3: Pasting Key to Authorize Client

Once Authorized the Client will now be able to connect to all endpoints listed on the Swagger UI document.

This method is best suited for backend services or trusted integrations where user-specific session management is not required.

2 Endpoints

This section provides an overview of the available API endpoints. The request and response samples shown here are for illustrative purposes only. For the most accurate and up-to-date data models, please refer to the Swagger UI documentation.

2.1 Authentication

2.1.1 Login

Authenticates a user and returns a JWT token for authorization.

URL: /api/auth/login

Method: POST

Authentication: None

Request Body:

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

Response :

```
{
  "token": "string"
}
```

Once the token is generated click Authorize on the top right and paste in the token once successful the Lock icon should appear to be locked.

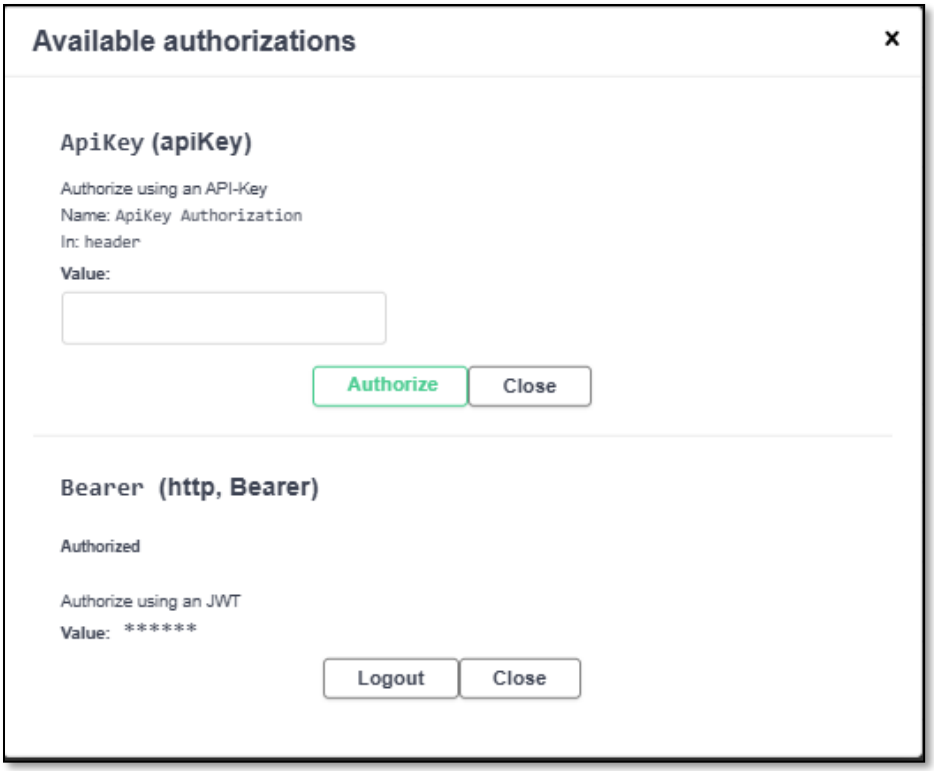

Figure 2.1: Successful Authorization using JWT

2.2 Job Requests

2.2.1 Get Job Requests

Returns a paginated list of job requests.

URL: /api/jobrequests

Method: GET

Authentication: Required

Parameters:

- page (integer, required): Page number
- pageSize (integer, required): Number of items per page
- orderBy (string, optional): Field to order by
- filter (string, optional): Filter criteria

Response: Paginated list of job requests

```
{
 "count": 1,
 "data": [
  {
   "jobRequestNumber": 1234,
   "details": "Request Details",
   "requestStatus": "Priority",
   "raisedBy": "John",
   "raisedByEmail": "John@mail.com",
        …
        }
  ]
}
```

2.2.2 Create a Job Request

Creates a new job request.

URL: /api/jobrequests

Method: POST

Authentication: Required

Request Body:

```
{
  "description": "string",
  "site": "string",
  "raisedBy": "string",
  "raisedByEmail": "string",
}
```

Response:

ID of the created job request

2.3 Jobs

2.3.1 Get Jobs

Returns a paginated list of jobs with optional filtering and ordering.

URL: /api/jobs

Method: GET

Authentication: Required

Parameters:

- page (integer, required): Page number
- pageSize (integer, required): Number of items per page
- orderBy (string, optional): Field to order by
- filter (string, optional): Filter criteria

Response:

```
{
 "count": 1,
 "data": [
  {
   "jobDetails": "Job Details",
   "jobNumber": 123,
   "raisedBy": "John",
   "reportedBy": "John@mail.com",
          …
          }
  ]
}
```

2.3.2 Get Job By ID

Returns details of a specific job.

URL: /api/jobs/{jobGuid}

Method: GET

Authentication: Required

Parameters:

- jobGuid (GUID, required): The unique identifier of the job

Response:

```
{
    "jobDetails": "Job Details",
    "jobNumber": 123,
    "raisedBy": "John",
    "reportedBy": "John@mail.com",
        ...
}
```

2.3.3 Create a Job

Creates a new job.

URL: /api/jobs

Method: POST

Authentication: Required

Request Body:

```
{
  "jobDetails": "string",
  "siteId": 0,
  "siteName": "string",
}
```

Response: GUID of the created job

"00000000-0000-0000-0000-000000000000"

2.3.4 Complete a Job

Marks a job as completed.

URL: /api/jobs/complete/{jobGuid}

Method: PUT

Authentication: Required

Parameters:

- jobGuid (GUID, required): The unique identifier of the job

Request Body:

```
{
  "jobId": 123,
  "completedNote": Note info,
  "completedByUserId": 123,
}
```

Response: GUID of the completed job

"00000000-0000-0000-0000-000000000000"

2.4 Job Costs

2.4.1 Get Job Costs

Returns a paginated list of costs associated with a job.

URL: /api/jobcosts/{jobGuid}

Method: GET

Authentication: Required

Parameters:

- jobGuid (GUID, required): The unique identifier of the job
- page (integer, required): Page number
- pageSize (integer, required): Number of items per page
- orderBy (string, optional): Field to order by
- filter (string, optional): Filter criteria

Response:


```
{
 "count": 1,
 "data": [
  {
   "jobDetails": 1
   "jobNumber":”1234-asdf-1234”,
   "invoiceNo": 12,
   "invoiceDate": "2025-04-16T07:29:08.737Z",
        }
  ]
}
```

2.4.2 Create Job Cost

Adds a cost entry to a job.

URL: /api/jobCosts/{jobGuid}

Method: POST

Authentication: Required

Parameters:

- jobGuid (GUID, required): The unique identifier of the job

Request Body:

```
{
  "costMaterials": 30,
  "costLabour": 34,
  "budget": 100,
  "costDescription": "description"
}
```

Response: GUID of the created Job Cost

"00000000-0000-0000-0000-000000000000"

2.4.3 Update Job Cost

Updates a job cost entry.

URL: /api/jobcosts/update/{jobCostGuid}

Method: PUT

Authentication: Required

Parameters:

- jobCostGuid (GUID, required): The unique identifier of the job cost

Request Body:

```
{
  "costMaterials": 30,
  "costLabour": 34,
  "budget": 100,
  "costDescription": "description"
}
```

Response: GUID of the created Job Cost

"00000000-0000-0000-0000-000000000000"

2.5 Job Tasks

2.5.1 Get Job Tasks

URL: /api/JobTasks

Method: GET

Authentication: Required

Parameters:

- page (integer, required): Page number
- pageSize (integer, required): Number of items per page
- orderBy (string, optional): Field to order by
- filter (string, optional): Filter criteria

Response:

```
{
 "count": 0,
 "data": [
  {
   "taskId": 0,
   "taskNumber": "string",
   "guid": "string",
   "jobNumber": 0,

   ....
  }
 ]
}
```

2.2.2 Get Job Task By ID

Returns details of a specific job task.

URL: /api/jobTasks/{jobTaskGuid}

Method: GET

Authentication: Required

Parameters:

- jobTaskGuid (GUID, required): The unique identifier of the job task

Response:

```
{
    "taskId": 0,
    "taskNumber": "string",
    "guid": "string",
    "jobNumber": 0,

    ....
}
```

2.2.3 Create a Job Task

Creates a new job task.

URL: /api/jobTasks

Method: POST

Authentication: Required

Request Body:

```
{
  "jobGuid": "string",
  "description": "string",
  "assignedToUserId": 0,
  "assignedUserName": "string",
  "durationInHours": 0,
  "resourceId": 0,
}
```

Response: GUID of the created job task

"00000000-0000-0000-0000-000000000000"

2.2.4 Start a Job Task

Updates a job task's status to started.

URL: /api/jobTasks/StartTask/{jobTaskGuid}

Method: PUT

Authentication: Required

Parameters:

- jobTaskGuid (GUID, required): The unique identifier of the job task

Request Body:

```
{
  "startDate": "2025-04-08T12:00:00Z"
}
```

2.2.5 Complete a Job Task

Updates a job task's status to completed.

URL: /api/jobTasks/CompleteTask/{jobTaskGuid}

Method: PUT

Authentication: Required

Parameters:

- jobTaskGuid (GUID, required): The unique identifier of the job task

Request Body:

```
{
  "completeDate": "2025-04-08T12:00:00Z",
  "completNote": "string"
}
```

2.2.6 Update a Job Task

Updates an existing job task.

URL: /api/JobTasks/Update/{jobTaskGuid}

Method: PUT

Authentication: Required

Parameters:

- jobTaskGuid (GUID, required): The unique identifier of the job task

Request Body:

```
{
  "description": "string",
  "assignedToUserName": "string",
  "priorityName": "string",
  "duration": 0,
  "expectedStartDate": "string",
  "expectedEndDate": "string",
  "startDate": "string",
  "completedDate": "string"
}
```

2.6 Assets

2.6.1 Get Assets

Returns a paginated list of assets.

URL: /api/assets

Method: GET

Authentication: Required

Parameters:

- page (integer, required): Page number
- pageSize (integer, required): Number of items per page
- orderBy (string, optional): Field to order by
- filter (string, optional): Filter criteria

Response:


```
{
 "count": 0,
 "data": [
  {
   "description": "string",
   "assetClassId": 10,
   "assetClass": "string",

   ....
  }
 ]
}
```

2.6.2 Get Asset By ID

Returns details of a specific asset.

URL: /api/assets/{assetGuid}

Method: GET

Authentication: Required

Parameters:

- assetGuid (GUID, required): The unique identifier of the asset

Response:

```
{
    "description": "string",
    "assetClassId": 10,
    "assetClass": "string",
}
```

2.6.3 Create an Asset

Creates a new asset.

URL: /api/assets

Method: POST

Authentication: Required

Request Body:

```
{
    "description": "string",
    "assetClassId": 10,
    "assetClass": "string",

    ....
}
```

Response: Asset Guid

"00000000-0000-0000-0000-000000000000"

2.6.4 Update an Asset

Updates an existing asset.

URL: /api/assets/{assetGuid}

Method: PUT

Authentication: Required

Parameters:

- assetGuid (GUID, required): The unique identifier of the asset

Request Body:

```
{
    "description": "string",
    "assetClassId": 10,
    "assetClass": "string",

    ....
}
```

Response: Asset Guid

"00000000-0000-0000-0000-000000000000"

2.6.5 Delete an Asset

Deletes an asset.

URL: /api/assets/{assetGuid}

Method: DELETE

Authentication: Required

Parameters:

- assetGuid (GUID, required): The unique identifier of the asset

2.7 Sites

2.7.1 Get Sites

Returns a paginated list of sites.

URL: /api/sites

Method: GET

Authentication: Required

Parameters:

- page (integer, required): Page number
- pageSize (integer, required): Number of items per page
- orderBy (string, optional): Field to order by
- filter (string, optional): Filter criteria

Response:


```
{
 "count": 0,
 "data": [
  {
   "name": "string",
   "code": 10,
   "region": "string",

   ....
  }
 ]
}
```

2.7.2 Get Site By ID

Returns details of a specific site.

URL: /api/sites/{siteGuid}

Method: GET

Authentication: Required

Parameters:

- siteGuid (GUID, required): The unique identifier of the site

Response:

```
{
    "name": "string",
    "code": 10,
    "region": "string",
  }
```

2.7.3 Create a Site

Creates a new site.

URL: /api/sites

Method: POST

Authentication: Required

Request Body:

```
{
    "name": "string",
    "code": 10,
    "region": "string",

    ....
}
```

Response: GUID of the created site

"00000000-0000-0000-0000-000000000000"

2.7.4 Update a Site

Updates an existing site.

URL: /api/sites/{siteGuid}

Method: PUT

Authentication: Required

Parameters:

- siteGuid (GUID, required): The unique identifier of the site

Request Body:

```
{
    "name": "string",
    "code": 10,
    "region": "string",

    ....
}
```

Response:

Updated site object

2.8 Resources

2.8.1 Get Resources

Returns a paginated list of resources.

URL: /api/resources

Method: GET

Authentication: Required

Parameters:

- page (integer, required): Page number
- pageSize (integer, required): Number of items per page
- orderBy (string, optional): Field to order by
- filter (string, optional): Filter criteria

Response:


```
{
 "count": 0,
 "data": [
  {
   "name": "string",
   "resourceType": "string",
   "emailAddress": "string",

   ....
  }
 ]
}
```

2.8.2 Get Resource By ID

Returns details of a specific resource.

URL: /api/resources/{resourceGuid}

Method: GET

Authentication: Required

Parameters:

- resourceGuid (GUID, required): The unique identifier of the resource

Response:

```
{
 "name": "string",
 "resourceType": "string",
 "emailAddress": "string",

 ....
}
```

2.8.3 Create a Resource

Creates a new resource.

URL: /api/resources

Method: POST

Authentication: Required

Request Body:

```
{
 "name": "string",
 "resourceType": "string",
 "emailAddress": "string",
```

```
  ....
}
```

Response: GUID of the created resource

"00000000-0000-0000-0000-000000000000"

2.8.4 Update a Resource

Updates an existing resource.

URL: /api/resources/{resourceGuid}

Method: PUT

Authentication: Required

Parameters:

- resourceGuid (GUID, required): The unique identifier of the resource

Request Body:

```
{
    "name": "string",
    "resourceType": "string",
    "emailAddress": "string",

    ....
}
```

Response: GUID of the created resource

"00000000-0000-0000-0000-000000000000"

2.9 Storage Areas

2.9.1 Get Storage Areas

Returns a paginated list of storage areas.

URL: /api/storageArea

Method: GET

Authentication: Required

Parameters:

- page (integer, required): Page number
- pageSize (integer, required): Number of items per page
- orderBy (string, optional): Field to order by
- filter (string, optional): Filter criteria

Response:


```
{
 "count": 0,
 "data": [
  {
   "storageName": "string",
   "code": "string",
   "siteId": 0,

   ....
  }
 ]
}
```

2.9.2 Get Storage Area By ID

Returns details of a specific storage area.

URL: /api/storageArea/{guid}

Method: GET

Authentication: Required

Parameters:

- guid (GUID, required): The unique identifier of the storage area

Response:

```
{
    "storageName": "string",
    "code": "string",
    "siteId": 0,

    ....
}
```

2.9.3 Create a Storage Area

Creates a new storage area.

URL: /api/storageArea

Method: POST

Authentication: Required

Request Body:

```
{
    "storageName": "string",
    "code": "string",
    "siteId": 0,

    ....
}
```

Response: GUID of the created storage area

"00000000-0000-0000-0000-000000000000"

2.9.4 Update a Storage Area

Updates an existing storage area.

URL: /api/storageArea/{guid}

Method: PUT

Authentication: Required

Parameters:

- guid (GUID, required): The unique identifier of the storage area

Request Body:

```
{
    "storageName": "string",
    "code": "string",
    "siteId": 0,

    ....
}
```

Response: GUID of the updated storage area

"00000000-0000-0000-0000-000000000000"

2.10 Stocks

2.10.1 Get Stocks

Returns a paginated list of stocks.

URL: /api/stocks

Method: GET

Authentication: Required

Parameters:

- page (integer, required): Page number
- pageSize (integer, required): Number of items per page
- orderBy (string, optional): Field to order by
- filter (string, optional): Filter criteria

Response:


```
{
 "count": 0,
 "data": [
  {
   "guid": "string",
   "stockId": 0,
   "stockTypeId": 0,

   ....
  }
 ]
}
```

2.10.2 Get Stock By ID

Returns details of a specific stock.

URL: /api/stocks/{guid}

Method: GET

Authentication: Required

Parameters:

- guid (GUID, required): The unique identifier of the stock

Response:

```
{
    "guid": "string",
    "stockId": 0,
    "stockTypeId": 0,

    ....
 }
```

2.10.3 Create a Stock

Creates a new stock.

URL: /api/Stocks

Method: POST

Authentication: Required

Request Body:

```
{
    "guid": "string",
    "stockId": 0,
    "stockTypeId": 0,

    ....
}
```

Response: GUID of the created stock

"00000000-0000-0000-0000-000000000000"

2.10.4 Update a Stock

Updates an existing stock.

URL: /api/stocks/{guid}

Method: PUT

Authentication: Required

Parameters:

- guid (GUID, required): The unique identifier of the stock

Request Body:

```
{
    "guid": "string",
    "stockId": 0,
    "stockTypeId": 0,

    ....
 }
```

Response:

"00000000-0000-0000-0000-000000000000"

2.11 Stock Levels

2.11.1 Get Stock Levels

Returns a paginated list of stock levels.

URL: /api/stockLevel

Method: GET

Authentication: Required

Parameters: -

- page (integer, required): Page number
- pageSize (integer, required): Number of items per page
- orderBy (string, optional): Field to order by
- filter (string, optional): Filter criteria

Response: Paginated list of stock levels


```
{
 "count": 0,
 "data": [
  {
   "stockLevelId": "string",
   "stockId": 0,
   "storageId": 0,

   ....
  }
 ]
}
```

2.11.2 Get Stock Level By ID

Returns details of a specific stock level.

URL: /api/stocklevel/{guid}

Method: GET

Authentication: Required

Parameters:

- guid (GUID, required): The unique identifier of the stock level

Response:

```
{
    "stockLevelId": "string",
    "stockId": 0,
    "storageId": 0,

    ....
 }
```

2.11.3 Update or Create a Stock Level

Updates an existing stock level or creates a new one if it doesn't exist.

URL: /api/stocklevel

Method: POST

Authentication: Required

Request Body:

```
{
    "stockLevelId": "string",
    "stockId": 0,
    "storageId": 0,
 }
```

Response: GUID of the created/updated stock level

"00000000-0000-0000-0000-000000000000"

3. Error Handling

The API uses standard HTTP status codes to indicate the success or failure of requests:

- 200 OK: The request was successful
- 201 Created: A new resource was successfully created
- 400 Bad Request: The request was malformed or invalid
  - Usually includes a response body with validation error details
- 401 Unauthorized: Authentication failed
  - Check that your authentication token is valid and properly formatted
- 403 Forbidden: The authenticated user does not have permission to access the requested resource
  - User may need additional role permissions
- 404 Not Found: The requested resource was not found
  - Check that the ID or GUID being used is correct
- 500 Internal Server Error: An error occurred on the server
  - Contact your TrackplanFM system administrator if this persists

3.1 Common Error Responses

Validation Errors (400 Bad Request)

```
{
 "errors": {
  "propertyName": [
   "Error message describing the validation failure"
 ]
 },
 "title": "One or more validation errors occurred.",
 "status": 400
}
```

Authentication Errors (401 Unauthorized)


```
{
  "type": "https://tools.ietf.org/html/rfc7235#section-3.1",
  "title": "Unauthorized",
  "status": 401,
  "detail": "Invalid authentication credentials"
}
```


Permission Errors (403 Forbidden)


```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.3",
  "title": "Forbidden",
  "status": 403,
  "detail": "Insufficient permissions to access the requested resource"
}
```

4 Filtering and Sorting

Many endpoints support filtering and sorting through the filter and orderBy query parameters.

4.1 Filtering

The filter parameter accepts a list of filter expressions:

Name	Operator	Usage
Equal	==	FieldName ==Value
NotEqual	!=	FieldName !=Value
LessThan	<	FieldName < Value
GreaterThan	>	FieldName > Value
GreaterThanOrEqual	>=	FieldName >= Value
LessThanOrEqual	<=	FieldName <= Value
Contains - Like	=*	FieldName =* Value
NotContains - NotLike	!*	FieldName !* Value
StartsWith	^	FieldName ^ Value
NotStartsWith	!^	FieldName !^ Value
EndsWith	$	FieldName $ Value
NotEndsWith	!$	FieldName !$ Value
AND - &&	,	FieldName , Value

4.2 Sorting

The orderBy parameter accepts a property name with optional direction:

- Ascending (default): propertyName or propertyName ASC
- Descending: propertyName DESC
- Multiple fields: propertyName1, propertyName2 DESC

Examples: - orderBy=name - orderBy=priority DESC - orderBy=region,name DESC

5 Best Practices

5.1 Efficient Querying

- Use filtering to narrow down results when retrieving lists of items
- Request only the data you need by limiting page sizes
- Use appropriate ordering to improve user experience

5.2 Error Handling

- Implement proper error handling for all API responses
- Always check status codes in responses
- Parse error messages to provide useful feedback to users
- Consider implementing retry logic with exponential backoff for temporary failures

5.3 Authentication

- Store JWT tokens securely
- Refresh tokens before they expire
- Never expose API keys in client-side code
- Implement proper token rotation for security

6 Support

For additional assistance, please contact your TrackplanFM administrator or support team