API interface

The application programming interface (API) is used to make a connection between MXSuite and other software.

- Manage API keys
- General API usage
- Update counters via API
- API for Assets Tasks

Manage API keys

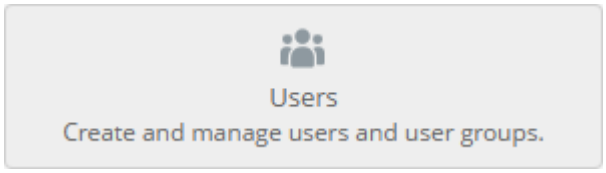
The HTTP(S) API has been available since MXSuite version 3.3.0.

To create an API key, there are 2 steps needed:

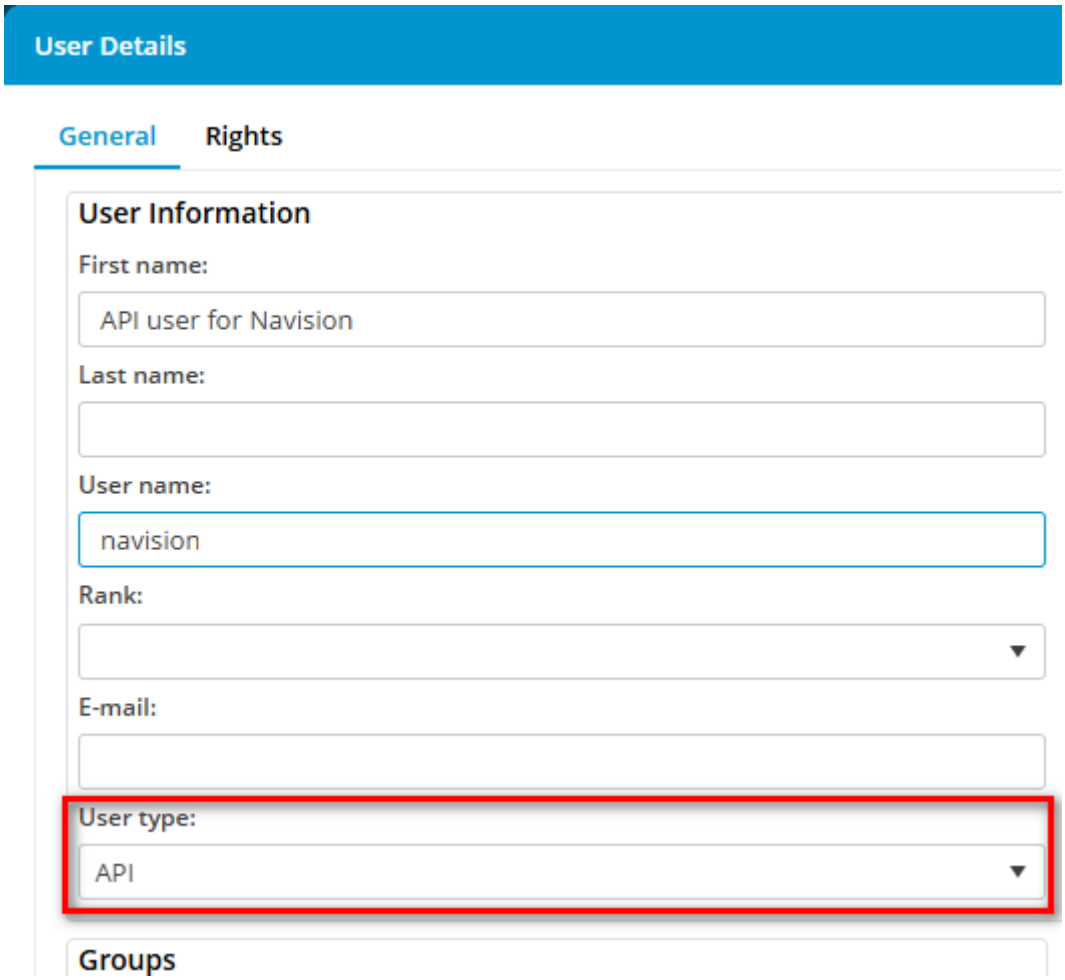
1. Create the API-user
2. Create the API-key

Create an API user

1. Open the **Administration** module
2. Click on **Users**

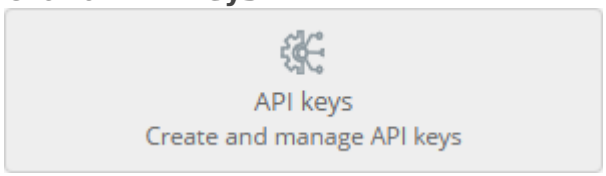


3. Create a new user for the API with the correct user rights. Assure that you select user type API.

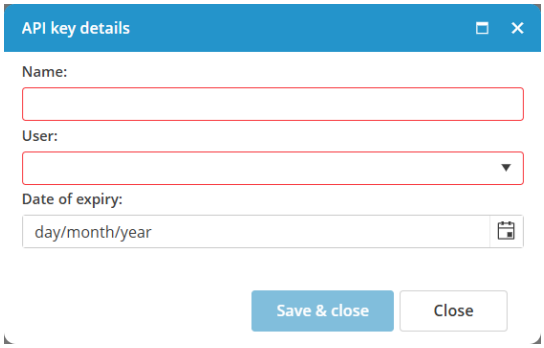


Create the API key

1. Open the **Administration** module
2. Click on **API keys**



3. Click on **New...**



4. Enter the **Name** of the API so you can easily distinguish all API keys.
5. Select the related API user
6. If needed, enter an expiry date.
7. Click on **Save & close**
8. Now the API key is created and visible.

General API usage

Authentication

To access the API, first an API key needs to be created from the Administration page. The key will be used as a header in the HTTP request.

MXSuite expects a `mx-apikey` header on each request.

```
mx-apikey: apikey
```

> Besides the mx-apikey header, two more headers are required.
>
> **accept**: application/json, text/plain, */*
> **content-type**: application/json

Error handling

Successful responses will return a 200 or 204 HTTP response code. Errors will return a 4xx or a 5xx.

Some scenarios of errors:

- missing the mx-apikey header
- wrong api key
- expired api key
- MXSuite license expired
- The user doesn't have enough rights for that specific request

Examples

Below an example of how to use the API key in MXSuite. In this example the following parameters are used:

- URL for MXSuite: http://localhost:4200/
- Location mane: vessel2
- API key: 4a74ad039ffc4f3288da7a0e03e608da

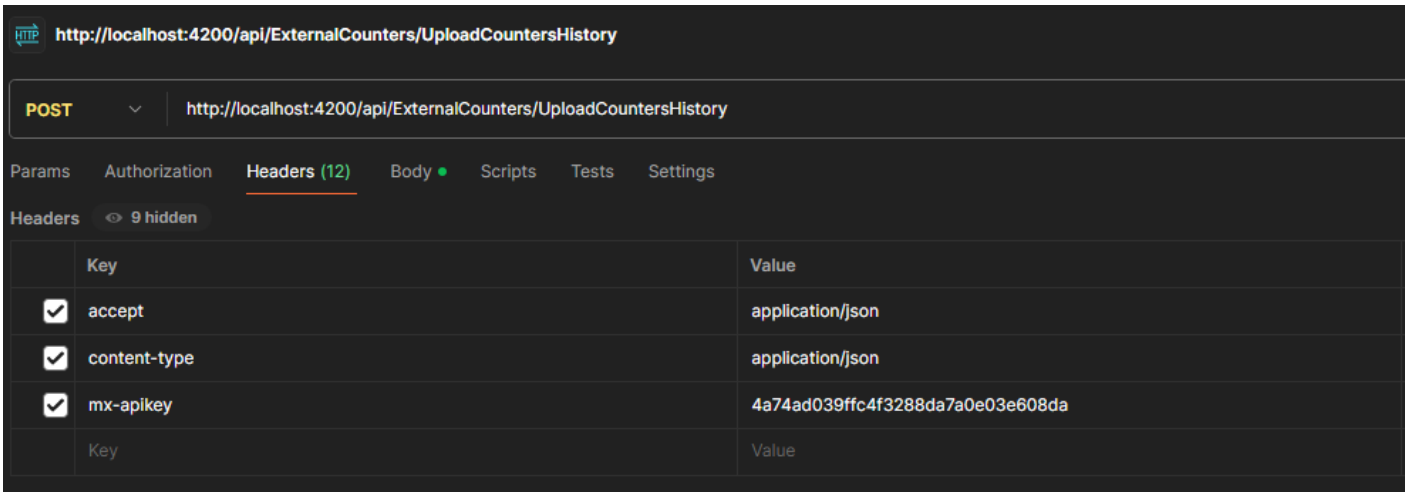
Postman

Postman can be downloaded here: https://www.postman.com/downloads/
Open Postman, click on Import button and paste the following command:

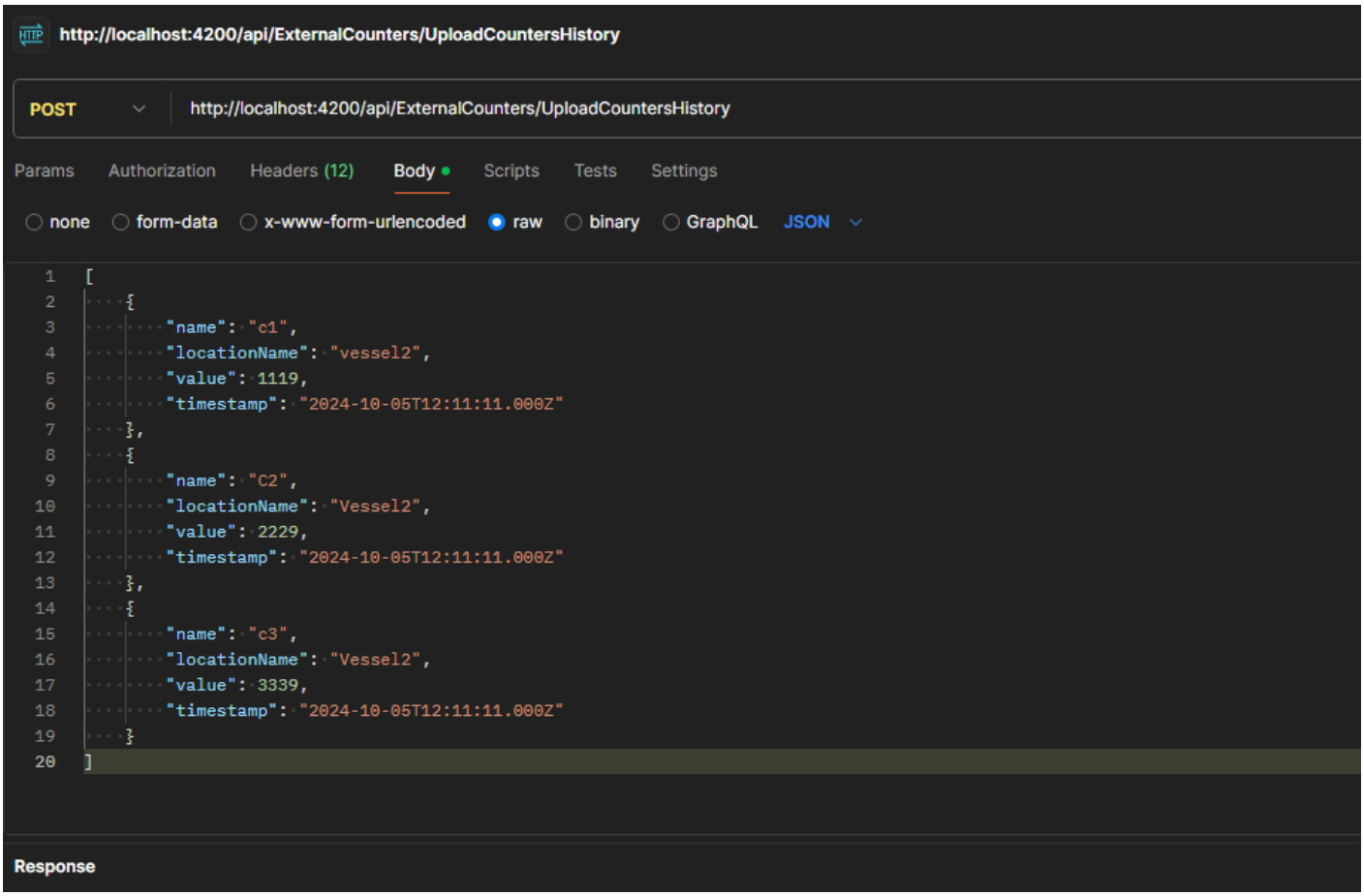
```
curl 'http://localhost:4200/api/ExternalCounters/UploadCountersHistory' \
  -H 'accept: application/json' \
```

```
  -H 'content-type: application/json' \
  -H 'mx-apikey: 4a74ad039ffc4f3288da7a0e03e608da' \
  --data-raw '[{"name":"c1","locationName":"vessel2","value":1119,"timestamp":"2024-10-05T12:11:11.000Z"},{"name":"C2","locationName":"Vessel2","value":2229,"timestamp":"2024-10-05T12:11:11.000Z"},{"name":"c3","locationName":"Vessel2","value":3339,"timestamp":"2024-10-05T12:11:11.000Z"}]'
```

In the Headers tab the API key can be changed:



In the Body tab, the payload value can be changed.



Console

Define a payload variable:

```
let payload = [
    {
        "name": "c1",
"locationName": "vessel2",
        "value": 1119,
        "timestamp": new Date(2024,09,05,15,11,11)
    },
    {
        "name": "C2",
"locationName": "Vessel2",
        "value": 2229,
        "timestamp": new Date(2024,09,05,15,11,11)
    },
    {
        "name": "c3",
"locationName": "Vessel2",
        "value": 3339,
        "timestamp": new Date(2024,09,05,15,11,11)
    }
];
```

Request:

```
fetch("http://localhost:4200/api/ExternalCounters/UploadCountersHistory", {
  "headers": {
    "accept": "application/json",
    "content-type": "application/json",
    "mx-apikey": "4a74ad039ffc4f3288da7a0e03e608da",
  },
  "body": JSON.stringify(payload),
  "method": "POST"
}).then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

> Don't forget to change the mx-apikey header value

Update counters via API

Overview

The Counters API provides endpoints for managing running hours (counters) within the MXSuite system. This API allows you to upload counter history records for one or more counters at a location, with proper authorization controls.

Base URL

- api/external/Counters/

1. Upload Counters History

Overview

The Upload Counters History endpoint allows authorized users to upload a batch of running hours (counter) history records for one or more counters at a specific location. This is typically used to record the latest readings for equipment counters.

Endpoint

POST /api/external/Counters/UploadCountersHistory

Authentication

All endpoints require authentication and appropriate user rights.
This endpoint requires the UpdateRunningHours right and the location must not be read-only.
Authorization is enforced via the MXApiPolicyAuthorize attribute.

Request

```
{
  name: string,
  locationName: string,
  value: number,
  timeStamp: Date
}
```

```
[
  {
    "Name": "Main Engine",
    "LocationName": "Neptune",
```

```
    "Value": 12345,
    "TimeStamp": "2025-06-23T14:00:00Z"
  },
  {
    "Name": "Generator 1",
    "LocationName": "Neptune",
    "Value": 6789,
    "TimeStamp": "2025-06-23T14:00:00Z"
  }
]
```


Error Responses

- 400 Bad Request: Invalid input data (e.g., missing required fields, invalid values).
- 401 Unauthorized: User is not authenticated.
- 403 Forbidden: User does not have the required rights or location is read-only.
- 500 Internal Server Error: An unexpected error occurred.

> If there is no location with that name, the entry is ignored.
> If there is no counter with that name for that location, the entry is ignored.
> If there is already an entry added for that counter in that location with the same value and timestamp, the entry is ignored.

API for Assets Tasks

Overview

The Assets APIs provides endpoints for managing assets categories, assets groups, assets tasks and sign off tasks within the MXSuite system. This API allows you to retrieve assets categories, assets groups and task sign offs, to create, update, delete and retrive assets tasks with proper authorization controls.

Base URL

- api/external/AssetsCategories/
- api/external/AssetsGroups/
- api/external/AssetsTasks/
- api/external/AssetsTaskSignOffs/

1. Assets Categories

Overview

The Assets Category API provides endpoints for managing assets categories within the MXSuite system. This API allows you to retrieve assets categories with proper authorization controls.

Base URL

api/external/AssetsCategories/

Authentication

All endpoints require authentication and appropriate user rights. The API uses token-based authentication through the MXApiPolicyAuthorize attribute.

Endpoints

1. Get categories

Endpoint:

GET /api/external/AssetsCategories/Get?locationName=Neptune

Request:

Query parameter:
- locationName=Neptune

Validations:

- **LocationName**: Required, max length 128.

Response:

```
[
  {
    "id": "b1a2c3d4-e5f6-7890-abcd-1234567890ab",
    "uniqueId": "CAT-001",
    "name": "Engine Room",
    "parentId": "c2b3a4d5-e6f7-8901-bcda-2345678901bc",
    "displayIndex": 1
  }
]
```

2. Assets Groups

Overview

The Assets GroupsAPI provides endpoints for managing assets groups within the MXSuite system. This API allows you to retrieve assets groups with proper authorization controls.

Base URL

api/external/AssetsGroups/

Authentication

All endpoints require authentication and appropriate user rights. The API uses token-based authentication through the MXApiPolicyAuthorize attribute.

Endpoints

1. Get groups

Endpoint:

GET /api/external/AssetsGroups/Get?categoryId=b1a2c3d4-e5f6-7890-abcd-1234567890ab

Request:

Query parameter:
- categoryId=b1a2c3d4-e5f6-7890-abcd-1234567890ab

Response:

```
[
  {
    "id": "d3e4f5a6-b7c8-9012-cdab-3456789012cd",
    "type": 1,
    "uniqueId": "GRP-001",
    "name": "Auxiliary Systems",
    "displayIndex":1
  }
]
```

Validations:

- **CategoryId**: Required, valid GUID

3. Assets Tasks

Overview

The Assets Task API provides endpoints for managing maintenance tasks within the MXSuite system. This API allows you to create, update, delete, and retrieve maintenance tasks with proper authorization controls.

```
{
  "GroupId":"3C160DF1-3A5C-406F-BAC6-00A4AB8C44A9",
  "GroupType":1,
  "UniqueId": "Task-001",
  "TaskName": "Oil Change - Updated",
  "Interval": 45,
  "IntervalType": 1,
  "DueDate": "2025-06-01T00:00:00Z",
  "IsCounterBased": false,
  "IsRemarkMandatory": true,
  "IsProject": false,
  "IsRecurrent": true,
```

```
  "IsFixedInterval": true,
  "IsAtServiceRequest": false,
  "IsDefect": false,
  "TaskDescription": "Updated description for oil change.",
  "MaxInterval": 90,
  "MaxIntervalType": 2,
  "DueCounters": 150,
  "Downtime": 3.0,
  "WarningInterval": 10,
  "WarningIntervalType": 1,
  "CounterName": "Main Engine",
  "DefaultEstimatedBudget": {
    "Amount": 600.0,
    "Currency": "USD"
  },
  "Ranks": ["Rank1", "Rank2"],
  "ApproverRanks": ["Approver1"],
  "RequiresApproval": true
}
```

Base URL

api/external/AssetsTasks/

Authentication

All endpoints require authentication and appropriate user rights. The API uses token-based authentication through the MXApiPolicyAuthorize attribute.

Endpoints

1. Create Maintenance Task

Creates a new maintenance task in the system.

Endpoint

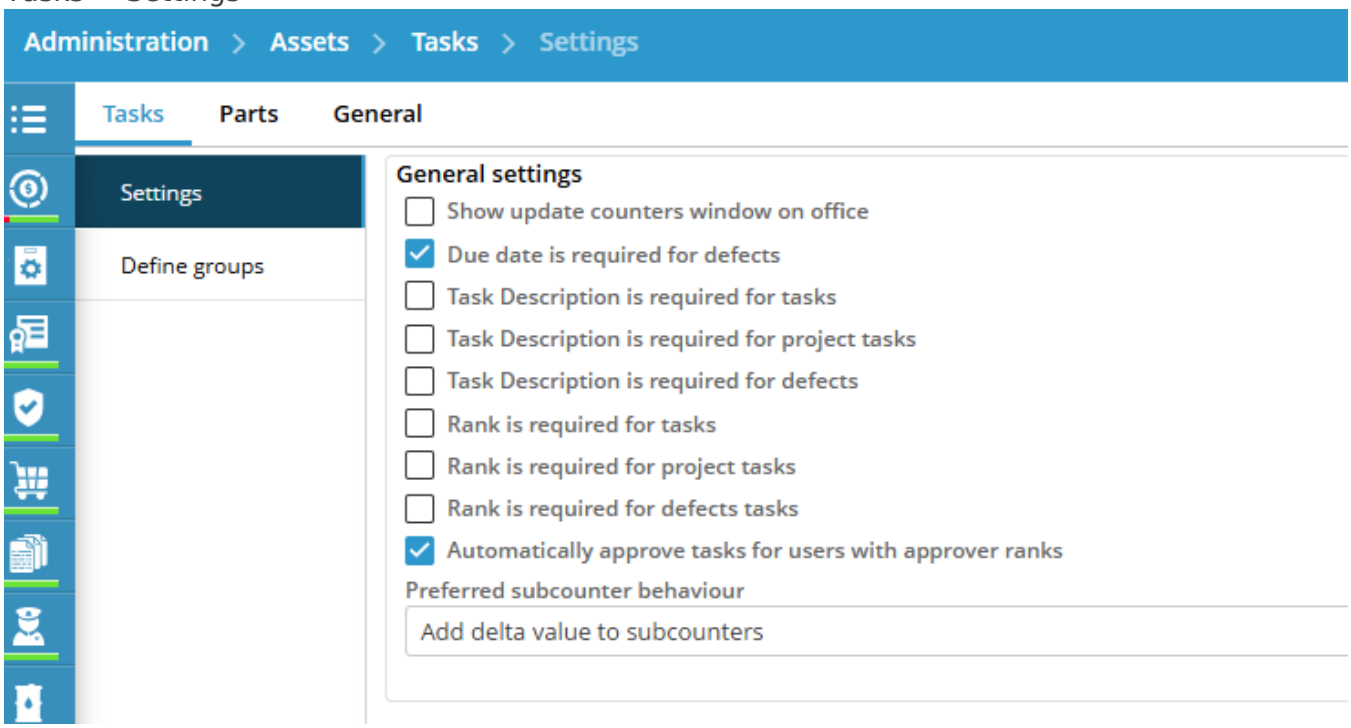
POST api/external/AssetsTasks/Create

Authorization Required:

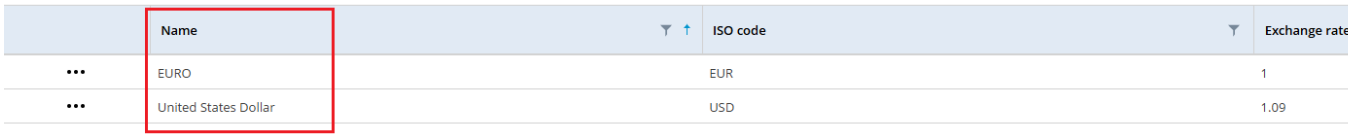
- User must have AddMaintenanceTask rights

Validations

- **GroupId:** identifies the group together with **GroupType.** Both are ***required***
- **UniqueId**: Max length 50. Not required, but should be unique if provided.
- **TaskName**: ***Required***, max length 128, must be unique.
- **Interval**: Must be greater than 0, within limits based on **IntervalType**.
- **IntervalType, MaxIntervalType** or **WarningIntervalType**: should contain one of these values:
  - 1 = days
  - 2 = weeks
  - 3 = months
- **DueDate**: Must be between **01/01/1900** and **01/01/2100**.
- **TaskDescription**: ***Required*** depending on how it is set in Administration > Assets > Tasks > Settings



- **Ranks**: ***Required*** for tasks requiring approval.
- **RequiresApproval**: Must be false for Silver licenses.
- **ApproverRanks:** ***Required*** only if **RequiresApproval** is set to **true.**
- **Currency**: when provided, it should contain the currency name as seen in Administration > Currencies



- When a **rank** is provided (e.g.: **Ranks** or **ApproverRanks**), the name from Administration > Ranks should be used:

	Name
	2nd Engineer
	2nd Engineer <3000KW
	A/B
	A/B Cook
	A/B Motorman

- **CounterName**: required when **IsCounterBased** is set to true. The values provided should be the ones defined in Symmary > Counters
- **WarningInterval + WarningIntervalType**: required when **IsCounterBased** is set to true.

Errors

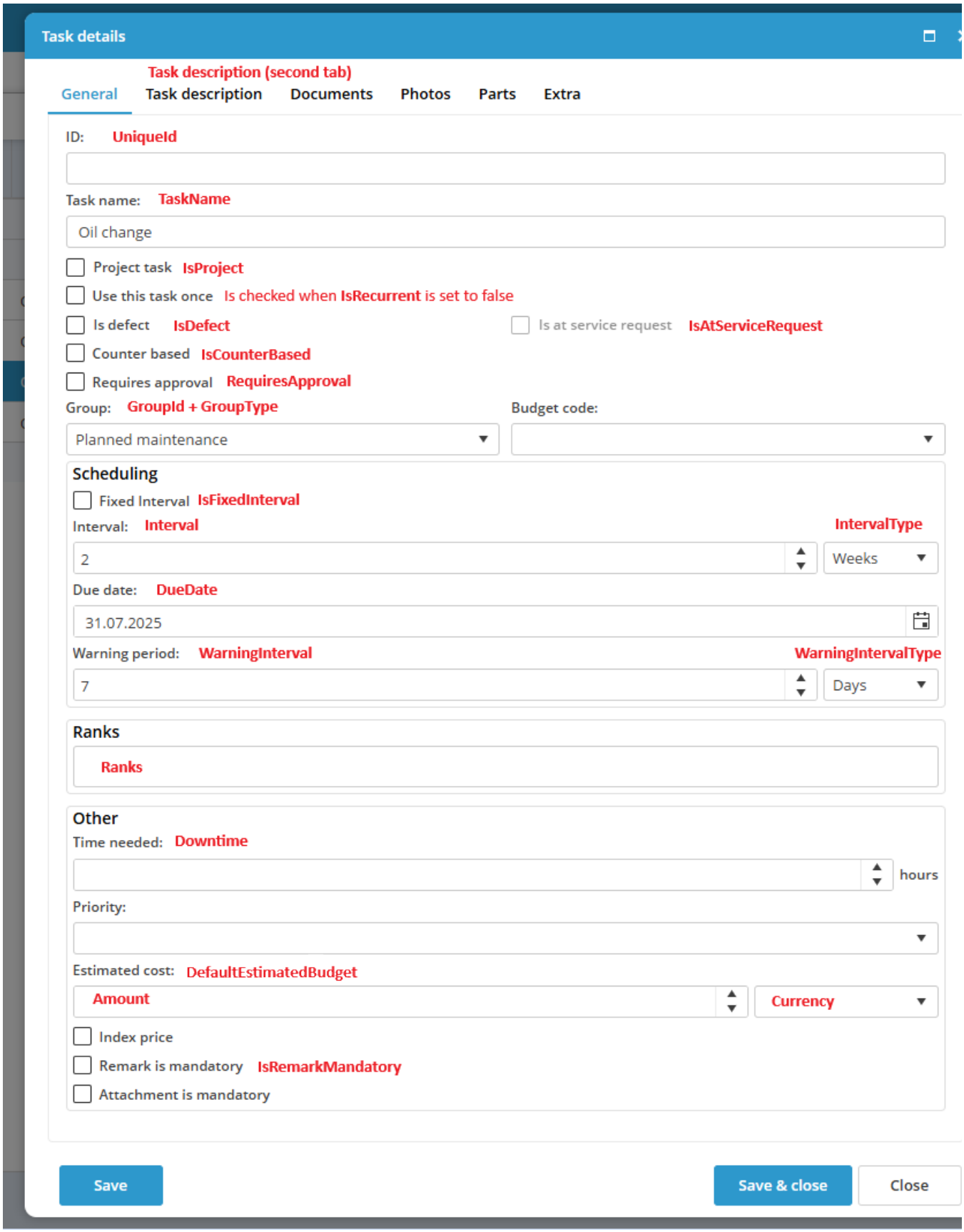
- 409 Conflict
- 404 Not Found: Task not found.
- 412 Precondition failed: Exceeding limits for intervals
- 403 Forbidden
- 400 Bad Request: Validation errors and required fields

Response

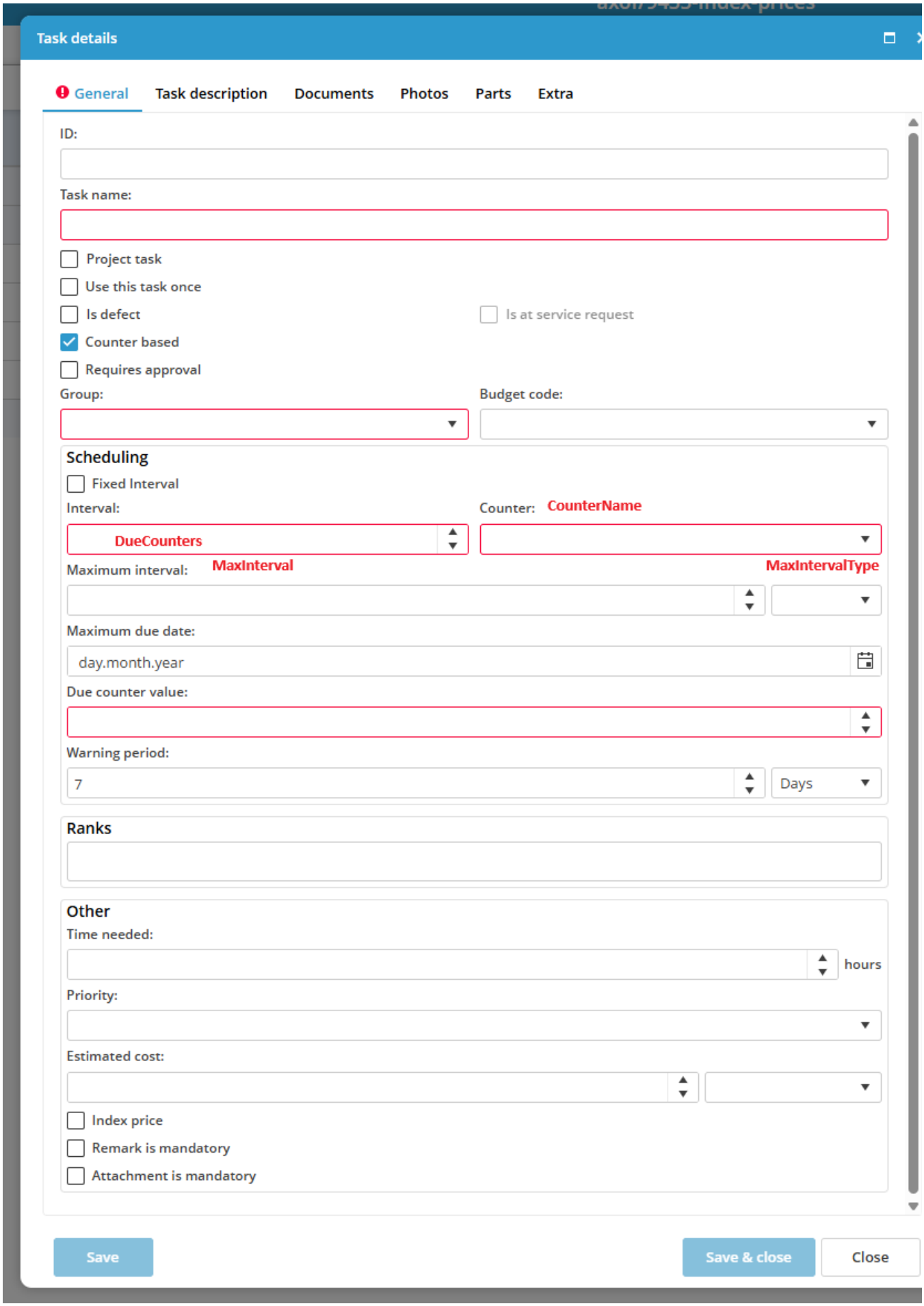
```
{
  "TaskId": "1463aea5-9062-45bf-8b9c-24cc5615d467"
}
```

Fields details:

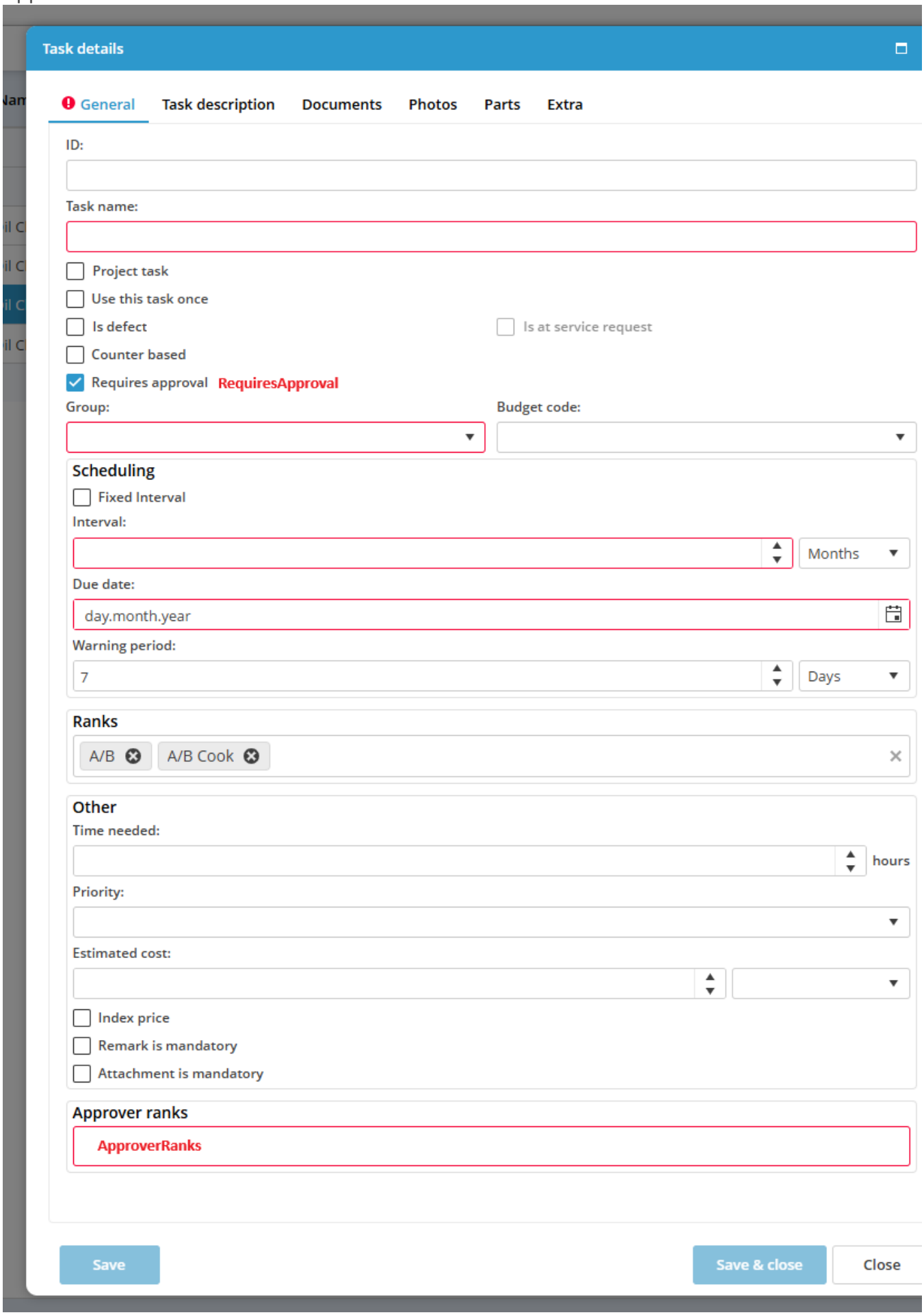
- General task details



- Counter based tasks:


Task details
General
Task description
Documents
Photos
Parts
Extra
ID:
Task name:
Project task
Use this task once
Is defect
Is at service request
Counter based
Requires approval
Group:
Budget code:
Scheduling
Fixed Interval
Interval:
Counter:
CounterName
DueCounters
Maximum interval:
MaxInterval
MaxIntervalType
Maximum due date:
day.month.year
Due counter value:
Warning period:
7
Days
Ranks
Other
Time needed:
hours
Priority:
Estimated cost:
Index price
Remark is mandatory
Attachment is mandatory
Save
Save & close
Close

- Approver ranks:



2. Update Maintenance Task

Updates an existing maintenance task.

Endpoint

PUT api/external/AssetsTasks/Update

Authorization Required

- User must have EditMaintenanceTask rights


```
{
  "UniqueId": "Task-001",
  "TaskName": "Oil Change - Updated",
  "TaskId": "1463aea5-9062-45bf-8b9c-24cc5615d467",
  "GroupId":"3C160DF1-3A5C-406F-BAC6-00A4AB8C44A9",
  "GroupType":1,
  "Interval": 45,
  "IntervalType": 1,
  "DueDate": "2025-06-01T00:00:00Z",
  "IsCounterBased": false,
  "IsRemarkMandatory": true,
  "IsProject": false,
  "IsRecurrent": true,
  "IsFixedInterval": true,
  "IsAtServiceRequest": false,
  "IsDefect": false,
  "TaskDescription": "Updated description for oil change.",
  "MaxInterval": 90,
  "MaxIntervalType": 2,
  "DueCounters": 150,
  "Downtime": 3.0,
  "WarningInterval": 10,
  "WarningIntervalType": 1,
  "CounterName": "Main Engine",
  "DefaultEstimatedBudget": {
    "Amount": 600.0,
    "Currency":  "EURO"
  },
  "Ranks": ["Rank1", "Rank2"],
```

```
  "ApproverRanks": ["Approver1"],
  "RequiresApproval": true
}
```

Validations

Same as the **Create** action.

Errors

- 409 Conflict
- 404 Not Found: Task not found.
- 412 Precondition failed: Exceeding limits for intervals
- 403 Forbidden
- 400 Bad Request: Validation errors and required fields

3. Delete Maintenance Task

Description: Deletes a maintenance task.

- **HTTP Method**: DELETE
- There can be to types of request bodies, please see examples below

Authorization Required

- User must have Delete Maintenance Task rights

Request body for URL /api/external/AssetsTasks/delete?taskId={taskId}

Response

- Status Code: 204 No Content
- Body: Empty response on success.

Validations

- **taskId:** Required, must be a valid GUID.

Errors

- 404 Not Found: Task not found.
- 500 Internal Server Error: Unexpected errors.

4. Get

Endpoint

/api/external/AssetsTasks/Get**?taskId={taskId}**

Validations

- taskId: Required, must be a valid GUID.

5. Get Multiple Tasks

Endpoint

/api/external/AssetsTasks/GetTasks

Request JSON

```
[
    "1463aea5-9062-45bf-8b9c-24cc5615d467",
    "7b95cb8b-0cb8-4cdf-ad3e-3285a162cbea"
]
```

Validations

- taskIds: Required, must be a list of valid GUIDs.

6. Get Tasks Ids

Retrieves the IDs of maintenance task/s.

- HTTP Method: GET
- URL: /api/external/AssetsTasks/GetTasksIds

Authorization Required

- Location must not be read-only

Request body

```
{
  "GroupId": "d3e4f5a6-b7c8-9012-cdab-3456789012cd",
  "GroupType": 1
}
```

Response

- Status Code: 200 OK

```
[
  "e4f5a6b7-c8d9-0123-dabc-4567890123de",
  "f5a6b7c8-d9e0-1234-abcd-5678901234ef"
]
```

Validations

- **GroupId**: Must be GUID• **GroupType**: Must be a positive number.

Errors

- 404 Not Found: Task not found.
- 500 Internal Server Error: Unexpected errors.

4. Assets Task Sign Offs

Overview

The Assets Task Sign Offs API provides endpoints for managing task sign offs within the MXSuite system. This API allows you to retrieve task sign offs with proper authorization controls.

Base URL

api/external/AssetsTaskSignOffs/

Authentication

All endpoints require authentication and appropriate user rights. The API uses token-based authentication through the MXApiPolicyAuthorize attribute.

Endpoints

1. Get Sign-Offs

Endpoint

/api/external/AssetsTaskSignOffs/Get

Request JSON:

```
{
  "TaskId": "7b95cb8b-0cb8-4cdf-ad3e-3285a162cbea",
```

```
  "NumberOfSignOffs": 5
}
```

Validations

- TaskId: Required, must be a valid GUID.
- NumberOfSignOffs: Optional, must be a valid number.

Authorization

- Ensure the user has the required rights (UserRights) for each endpoint (Add task, edit task, delete task)
- The LocationIsNotReadOnly flag must be true for all operations.

Error Responses

All error responses follow this structure:

```
{
  "Errors": [
    {
      "message": "string",
      "errorCode": number,
      "isBusinessException": false,
      "businessExceptionData": null,
      "failureServerReason": null
    }
  ]
}
```

```
{
    "errors": [
        {
            "message": "Category name is required",
            "errorCode": 9,
            "isBusinessException": false,
            "businessExceptionData": null,
            "failureServerReason": null
        }
    ]
```

```
}
```

The API may return the following error responses:

1. 400 Bad Request
   - When request validation fails
   - When required fields are missing
   - When field values are invalid
1. 401 Unauthorized
   - When authentication token is missing or invalid
   - When user doesn't have required rights
1. 403 Forbidden
   - When location is read-only
   - When user doesn't have sufficient permissions
1. 404 Not Found
   - When requested task doesn't exist
   - When location or category doesn't exist
1. 409 Conflict
   - When trying to create a duplicate task
   - When task name already exists in the specified location/category/group

Validation Rules and status codes

0: **UnknownError** - A generic or unspecified error. Used as a fallback when the specific cause is not known.
Status: **400 Bad Request**

1–3: **Duplicate/Uniqueness Errors**
- 1: **TaskNameDuplicate**: The task name already exists in the system.
- 2: **UniqueIdDuplicate**: The unique identifier for a task is already in use.

- 3: **TaskNameDuplicateInGroup**: The task name is duplicated within a specific group.

Status: **409 Conflict**

4–10: **Not Found Errors**
- 4: **LocationNotFound**: The specified location does not exist.
- 5: **CategoryNotFound**: The specified category does not exist.
- 6: **GroupNotFound**: The specified group does not exist.
- 7: **TaskNotFound**: The specified task does not exist.
- 8: **RankNotFound**: The specified rank does not exist.
- 9: **ApproverRankNotFound**: The specified approver rank does not exist.
- 10: **CurrencyNotFound**: The specified currency does not exist.

Status: **404 Not Found**

11–31: **Required Field Errors**
- 11: **LocationRequired**: Location is required but missing.
- 12: **CategoryRequired**: Category is required but missing.
- 13: **GroupRequired**: Group is required but missing.
- 14: **TaskRequired**: Task is required but missing.
- 15: **TaskNameRequired**: Task name is required but missing.
- 16: **TaskDescriptionRequired**: Task description is required but missing.
- 17: **UniqueIdRequired**: Unique ID is required but missing.
- 18: **IntervalRequired**: Interval value is required but missing.
- 19: **IntervalTypeRequired**: Interval type is required but missing.
- 20: **WarningIntervalRequired**: Warning interval is required but missing.
- 21: **WarningIntervalTypeRequired**: Warning interval type is required but missing.
- 22: **MaxIntervalTypeRequired**: Max interval type is required but missing.
- 23: **ProjectRanksRequired**: Project ranks are required but missing.
- 24: **DefectRanksRequired**: Defect ranks are required but missing.
- 25: **ApproverRanksRequired**: Approver ranks are required but missing.
- 26: **RunningHoursRequired**: Running hours are required but missing.
- 27: **DueDateRequired**: Due date is required but missing.
- 28: **TaskIdRequired**: Task ID is required but missing.
- 29: **RankRequired**: Rank is required but missing.
- 30: **RanksRequired**: Ranks are required but missing.
- 31: **TaskListEmptyOrNull**: The list of tasks is empty or null.

Status: **400 Bad Request**

32–34: **Length Validation Errors**
- 32: **TaskNameTooLong**: Task name exceeds the maximum allowed length.
- 33: **LocationNameTooLong**: Location name exceeds the maximum allowed length.
- 34: **UniqueIdTooLong**: Unique ID exceeds the maximum allowed length.

Status: **400 Bad Request**

35–44: **Numeric/Value Validation Errors**
- 35: **GroupTypeGreaterThanZero**: Group type must be greater than zero.

- 36: **IntervalGreaterThanZero**: Interval must be greater than zero.
- 37: **WarningIntervalGreaterThanZero**: Warning interval must be greater than zero.
- 38: **MaxIntervalGreaterThanZero**: Max interval must be greater than zero.
- 39: **IntervalExceedsLimit**: Interval exceeds the allowed limit.
- 40: **WarningIntervalExceedsLimit**: Warning interval exceeds the allowed limit.
- 41: **MaxIntervalExceedsLimit**: Max interval exceeds the allowed limit.
- 42: **EstimatedBudgetAmountGreaterThanZero**: Estimated budget must be greater than zero.
- 43: **DueRunningHoursPositiveError**: Due running hours must be positive.
- 44: **DowntimeGreaterThanZero**: Downtime must be greater than zero. Status:
- 39–41: **412 Precondition Failed**
- Others: **400 Bad Request**

45–51: **Business Rule Validation Errors**
- 45: **ProjectCannotBeCounterBased**: Projects cannot be based on counters.
- 46: **TaskCompletionApprovalNotAllowed**: Task completion approval is not allowed.
- 47: **DefectRequiresOneTimeTask**: Defect tasks must be one-time tasks.
- 48: **ApproverRankNotAllowed**: Approver rank is not allowed.
- 49: **DowntimeNotAllowed**: Downtime is not allowed.
- 50: **AttachmentNotAllowed**: Attachments are not allowed.
- 51: **PriorityNotAllowed**: Priority is not allowed. Status:
- 45, 47: **412 Precondition Failed**
- 46, 48–51: **403 Forbidden**

54–56: **Invalid/Out-of-Range Errors**
- 54: **WarningIntervalTypeInvalid**: Warning interval type is invalid.
- 55: **IntervalTypeInvalid**: Interval type is invalid.
- 56: **MaxIntervalTypeInvalid**: Max interval type is invalid.
- 57: **InvalidRanks**: Ranks are invalid

Status: **400 Bad Request**

- 58: **UserIsNotAllowedOnLocation**: User is not allowed on the specified location (**400 Bad Request**)