IFS Aviation Maintenance

Technical Records Management API Guide

ICAM 26.2



Document Revision: 26.2-1

Publication Date: November 02, 2026

Contents

1 Introduction....3
Technical Records Management API overview....3

2 Setting up the Technical Records Management API....5
Licensing and subscription....5
Permissions and authentication....5
Assign the Technical Records Management integration role to a user....6
Give Technical Records Management integration permission to a user role....6

3 Technical Records Management API scenarios....8
Searching for a task by barcode....8
Searching for a technical reference by task ID....13
Adding an adhoc technical reference....14

Appendix A Error Handling....15
Logging....16

1

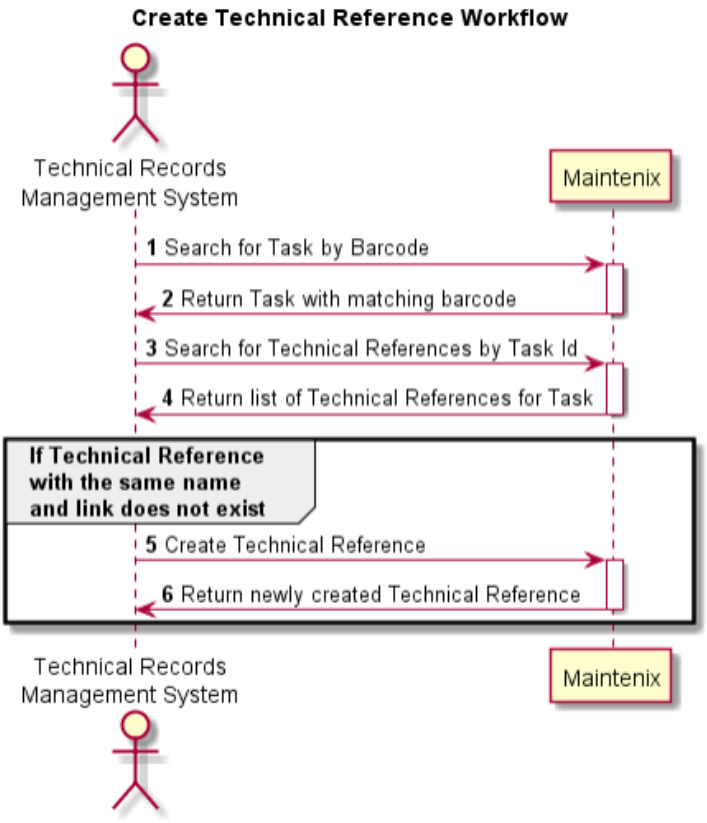
Introduction

Technical Records Management API overview

The Technical Records Management APIs enable integration with a third-party technical records management system.

The Technical Records Management APIs allow technical records management systems to associate a technical reference to a task or fault. Maintenix provides to the technical records management system the capability to publish a hyperlink—for example, a scanned maintenance record— to a task, or a corrective task associated with a fault. Tasks are identified by an Maintenix barcode.

The diagram below depicts a high-level overview of the Technical Records Management API integration.



A technical records management system (API client) integrates with the Technical Records Management API by sending a series of requests to the API endpoints. The first request is to search for a task by barcode. The second request is to use the task ID from the returned task to search for any technical references that belong to that task. The API client searches the list of technical references for a specific technical reference. If one is not found based on name and link, the API client sends a third request to create a new technical reference associated to that specific task.

Note If a timeout occurs, a client must send another search for technical references to determine if the create technical reference request succeeded. Repeated created requests can introduce duplicate technical references as there is no constraint on the name or URL. It is important to confirm a successful create technical reference request, before repeating the attempt, to avoid the potential for duplication.

2

Setting up the Technical Records Management API

After installing Maintenix, you set up and configure the Technical Records Management APIs by ensuring the proper license and integration role is available in your product.

The integration user role contains the required permissions You can assign this role to specific users in Maintenix. You can also give Technical Records Management API permissions directly to a different user role.

For information on how to install Maintenix, see the *Operator Installation and Configuration Guide*.

Licensing and subscription

To use the technical records management system, your organization must hold a valid license for it.

That means the `TECHNICAL_RECORDS_MANAGEMENT_API` feature must have been added to your Maintenix `license.xml`. If your current license does not include this feature, contact your Maintenix representative.

> **Note** `license.xml`, resides in the `<install-dir>/domain/config/maintenix` folder, on the Maintenix application server host.

Permissions and authentication

The technical records management integration role handles your permissions.

The Maintenix installation automatically creates a default user called `trmintegration` with the TRMINTEG role, which gives the user authorization to make requests to the Technical Records Management APIs in Maintenix. Any other user created for this purpose should be assigned the TRMINTEG role.

The TRMINTEG role is assigned the following permission set, which includes all the necessary permissions to invoke the Technical Records Management APIs:

- API_TASK_PARM
- API_TASK_TECHREF_PARM

After installing the Technical Records Management API, you can grant specific users permission to use the TRMINTEG role, or grant different user roles permission to use this API.

Assign the Technical Records Management integration role to a user

You can grant specific users permission to use a technical records management service.

Steps

1. Go to the **Role Search** page.
2. Enter TRMINTEG in the **Role Code** field and click **Search**.
3. On the **Role Details** page, **Users** tab, click **Assign User**.

> **Tip** If the user already appears on the **Role Details** page, select the user from the list and click **Assign User**. Other wise, follow the steps below.

4. On the **User Search** page, do any of the following:
   - Click **Search** to see a list of all users in the system.
   - Enter the complete or partial name of the user in any of the user name fields and click **Search**.
5. Select a user from the list.
6. Click **Assign User**.

Give Technical Records Management integration permission to a user role

You can grant users, with a specific role, permission to use a technical records management service.

Steps

1. Go to **Role Search** page.
2. On the **Role Search** page, do any of the following:
   - Click **Search** to see a list of all roles in the system.
   - Enter the code in the **Role Code** field and click **Search**.
   - Enter the complete or partial name of the role in the **Name** field and click **Search**.
3. Select a role from the list.
4. On the **Role Details** page, click the **Permission Sets** tab.

5. Under the **Maintenix API Extensions** setting, select the **Technical Records Management API** check box.
6. Click **OK**.

3

Technical Records Management API scenarios

The Technical Records Management APIs consists of calls to two different APIs—the Task API and Technical Reference API— which are used for associating a technical record with a task or fault in Maintenix.

The API calls can be described in distinct scenarios. These scenarios are:

- Searching for a task by barcode.
- Searching for a technical reference by task ID.
- Adding an adhoc technical reference.

Searching for a task by barcode

You send a request to the task search endpoint to return the corresponding task with the specified barcode.

The GET method's HTTP endpoint must contain a valid barcode.

Request

Endpoint: `http://<server:port>/amapi/maintenance/exec/task?barcode=<barcode>`

Response

A successful response consists of the following payload. You must find the task ID, located in the `id` value. This value is necessary for retrieving the technical reference.

```
{
  "issueToAccountId": "string",
  "originatorCode": "string",
  "minimumPlanningYield": 0,
  "reasonForRemovalCode": "string",
```

```
  "name": "string",
  "actions": [
    {
      "description": "string",
      "updatedDate": "2019-07-16T16:40:51.567Z"
    }
  ],
  "priority": "string",
  "id": "string",
  "status": "string",
  "creationDate": "2019-07-16T16:40:51.567Z",
  "description": "string",
  "usageSnapshots": [
    {
      "name": "string",
      "id": "string",
      "description": "string",
      "externalKey": "string",
      "inventoryId": "string",
      "documentRef": "string",
      "usageData": [
        {
          "id": "string",
          "usageParameter": "string",
          "tsn": 0,
          "tsi": 0,
          "tso": 0
        }
      ],
      "recordDate": "2019-07-16T16:40:51.567Z"
    }
  ],
  "scheduledStartDate": "2019-07-16T16:40:51.567Z",
  "drivingDeadline": {
    "deadlineType": "string",
    "extensionAmount": 0,
    "usageUntilDeadline": 0,
    "dataTypeCode": "string",
    "extendedDeadlineDate": "2019-07-16T16:40:51.571Z",
    "deadlineDate": "2019-07-16T16:40:51.571Z"
  },
  "duration": "string",
  "labour": [
    {
      "id": "4BF7D58C171A4517BF17DC5916E9F285",
      "taskId": "4BF7D58C171A4517BF17DC5916E9F285",
      "skillCode": "LBR",
      "stageCode": "ACTV",
      "workPerformed": {},
      "certificationRequired": false,
      "certification": {},
      "independentInspectionRequired": true,
      "independentInspection": {}
    }
  ],
```

```
  "historic": true,
  "barcode": "string",
  "technicalReference": [
    {
      "name": "string",
      "id": "string",
      "typeCode": "string",
      "number": "string",
      "url": "string",
      "description": "string",
      "inventoryId": "string",
      "note": "string"
    }
  ],
  "subclass": "string",
  "locationId": "string",
  "workTypes": [
    "string"
  ],
  "externalKey": "string",
  "inventoryId": "string",
  "taskCode": "string",
  "endDate": "2019-07-16T16:40:51.572Z",
  "parentId": "string",
  "taskClass": "string",
  "taskDefnId": "string",
  "measurements": [
    {
      "value": "string",
      "parameterDescription": "string",
      "inventoryId": "string",
      "unitOfMeasure": "string",
      "parameterCode": "string",
      "updatedDate": "2019-07-16T16:40:51.572Z"
    }
  ],
  "actualStartDate": "2019-07-16T16:40:51.572Z",
  "workPackageId": "string",
  "configSlot": "string",
  "panels": [
    {
      "labourSkill": "string",
      "status": "string",
      "maintenanceTaskId": "string",
      "labourHours": 0,
      "id": "string",
      "code": "string",
      "description": "string",
      "zoneCode": "string",
      "openPanel": {
        "date": "string",
        "accomplishedBy": "string",
        "skill": "string"
      },
      "closePanel": {
```

```
        "date": "string",
        "accomplishedBy": "string",
        "skill": "string"
      }
    }
  ],
  "zones": [
    {
      "id": "string",
      "code": "string",
      "description": "string"
    }
  ],
  "instructions": "string",
  "jobCardSteps": [
    {
      "statusCode": "string",
      "description": "string",
      "orderNumber": 0,
      "labourSteps": [
        {
          "notes": "string",
          "orderNumber": 0
        }
      ],
      "updatedDate": "2019-07-16T16:40:51.573Z"
    }
  ],
  "etopsSignificant": true,
  "taskWorkOrderLineNo": "string",
  "workPackageCommitLineOrder": 0,
  "scheduledEndDate": "2019-07-16T16:40:51.573Z",
  "statusDescription": "string",
  "partRequirements": [
    {
      "id": "9EEE1A1DA50F43189C806CC3D6D67BC2",
      "taskId": "9EEE1A1DA50F43189C806CC3D6D67BC2",
      "note": "string",
      "quantity": 0,
      "requestActionCode": "string",
      "assemblyPositionId": "9EEE1A1DA50F43189C806CC3D6D67BC2",
      "positionId": "9EEE1A1DA50F43189C806CC3D6D67BC2",
      "partGroupId": "9EEE1A1DA50F43189C806CC3D6D67BC2",
      "installedParts": [
        {
          "key": "string",
          "partId": "9EEE1A1DA50F43189C806CC3D6D67BC2",
          "inventoryId": "9EEE1A1DA50F43189C806CC3D6D67BC2",
          "serialBatchNumber": "string",
          "signedByUserId": "9EEE1A1DA50F43189C806CC3D6D67BC2",
          "quantity": 0
        }
      ],
      "removedParts": [
        {
```

```
          "key": "string",
          "partId": "9EEE1A1DA50F43189C806CC3D6D67BC2",
          "inventoryId": "9EEE1A1DA50F43189C806CC3D6D67BC2",
          "serialBatchNumber": "string",
          "reasonCode": "string",
          "signedByUserId": "9EEE1A1DA50F43189C806CC3D6D67BC2",
          "quantity": 0
        }
      ]
    }
  ],
  "parentTaskClass": "string",
  "toolRequirements": [
    {
      "signedBy": "string",
      "availableLocal": "string",
      "scheduledHours": 0,
      "availableRemote": "string",
      "partId": "string",
      "inventoryId": "string",
      "actualHours": 0,
      "baselined": true,
      "checkedOutTo": "string"
    }
  ],
  "schedulePriorityCode": "string",
  "parentAssemblyId": "string",
  "warrantyNote": "string",
  "softDeadline": true,
  "revisionDate": "2019-07-16T16:40:51.573Z",
  "taskClassMode": "string",
  "taskType": "Fault",
  "faultSeverity": "string",
  "userSubclass": "string",
  "foundFaultIds": [
    "string"
  ],
  "partGroupId": "string",
  "impacts": [
    {
      "code": "string",
      "name": "string",
      "description": "string"
    }
  ]
}
```

Note Although there are many fields available in this API's response, technical records management integrations should limit using or parsing fields other than `id` and `barcode`. This is because many properties currently available in the response might be deprecated or removed in the future, causing your integration to break. Please contact your Maintenix account representative if you find the need to use additional properties, so that we can ensure they are preserved in future versions of Maintenix.

Searching for a technical reference by task ID

You send a request to the technical reference search endpoint, using the retrieved task ID, to locate an associated technical reference.

The GET method's HTTP endpoint must contain a valid task ID. This ID is retrieved by the GET `/maintenance/exec/task` method. A 404 error is returned if you do not provide a valid task ID.

Request

Endpoint: `http://<server:port>/amapi/maintenance/exec/task/technicalreference?task_id=<task_id>`

Response

A successful response consists of the following payload. Note the following technical reference parameters:

- `name`: the technical reference name.
- `taskId`: the associated task ID.

If an associated technical reference is not found, a technical reference name and task ID must be provided in a POST method to create a new associated technical reference.


```
[
    {
        "key": "e0FFU31FRXU3cllUVldjRmtrZEcxOEl1Ry9nPT0=",
        "taskId": "07BA435652E4451A960CBFBCC5572694",
        "ietmTechnicalReferenceKey": null,
        "name": "dhdhhhdhhhr",
        "link": "<website>",
        "typeCode": "PDF",
        "description": "Test Description",
        "notes": "Test Notes",
        "printable": false,
        "applicabilityRange": null
    }
]
```

Adding an adhoc technical reference

If an associated technical reference cannot be found, you can send a request to the technical reference endpoint to create a technical reference associated to the specific task.

This POST method requires a task ID and technical reference name. The GET method, described in the scenario above, is used to search for technical references by task ID.

Request

Endpoint: `http://<server:port>/amapi/maintenance/exec/task/technicalreference`

POST - Add a new adhoc technical reference.

```
{
  "taskId": "2AEEFC45E5D411E4A577C190F489A76A",
  "ietmTechnicalReferenceKey": null,
  "name": "dhdhhhdhhhr",
  "link": "file:///c:/temp/pdfs/test.pdf",
  "typeCode": "PDF",
  "description": "Test Description",
  "notes": "Test Notes",
  "printable": true
}
```

Response

A successful response consists of the following payload, indicating that a technical reference has been created and linked to the task.

```
[
    {
        "key": "b0FFU22FRXU3d33UVldjRmtrZRcxOEl1Ry8mPT0=",
        "taskId": "2AEEFC45E5D411E4A577C190F489A76A",
        "ietmTechnicalReferenceKey": null,
        "name": "dhdhhhdhhhr",
        "link": "file:///c:/temp/pdfs/test.pdf",
        "typeCode": "PDF",
        "description": "Test Description",
        "notes": "Test Notes",
        "printable": true,
        "applicabilityRange": null
    }
]
```

A

Error Handling

When an HTTP request is made, a status code is included in the response, indicating a successful or unsuccessful reply.

Additional information can be found in the Maintenix logs.

Successful request

In the event of a successful request, Maintenix returns the following status code:

Status Code	Message	Description
200	OK	Standard successful HTTP request

In the event of a business error, the server logs indicate the following error code and a description:

Status Code	Message	Description
400	Bad Request	The request does not meet the API specification.
401	Unauthorized	The user is not authenticated.
403	Forbidden	The authenticated user does not have the necessary permissions for the resource.
404	Not Found	The resource could not be found

In the event of a technical error, the server logs indicate the following error code and a description:

Status Code	Message	Description
500	Internal Server Error	The request could not be processed due to an internal server error.

Logging

Integration events, exceptions, and responses are logged in the Maintenix log files. The log files use standard formats that you can configure to work with your own log processing and monitoring tools.

The primary log file is `maintenix.log`. This file uses Log4j format.

HTTP requests are also logged in `weblogic-access.log` which you have the option to monitor in real time.

For information on configuring logging, see the *Maintenix Installation and Configuration Guide*.

ABOUT IFS

IFS develops and delivers enterprise software for customers around the world who manufacture and distribute goods, maintain assets, and manage service-focused operations. The industry expertise of our people and solutions, together with commitment to our customers, has made us a recognized leader and the most recommended supplier in our sector. Our team of 3,500 employees supports more than 10,000 customers world-wide from a network of local offices and through our growing ecosystem of partners.

#forthechallengers

ifs.com

WHERE WE ARE

AMERICAS

+1 888 437 4968

ASIA PACIFIC

+65 63 33 33 00

EUROPE EAST

+48 22 577 45 00

EUROPE CENTRAL

+49 9131 77 340

UK & IRELAND

+44 1494 428 900

FRANCE, BENELUX AND IBERICA

+33 3 89 50 72 72

MIDDLE EAST AND AFRICA

+971 4390 0888

NORDICS

+46 13 460 4000

COPYRIGHT © 2026 INDUSTRIAL AND FINANCIAL SYSTEMS, IFS AB. IFS AND ALL IFS PRODUCTS AND SERVICES NAMES ARE TRADEMARKS OF IFS. ALL RIGHTS RESERVED. THIS DOCUMENT MAY CONTAIN STATEMENTS OF POSSIBLE FUTURE FUNCTIONALITY FOR IFS'S PRODUCTS AND TECHNOLOGY. SUCH STATEMENTS ARE FOR INFORMATION PURPOSES ONLY AND SHOULD NOT BE INTERPRETED AS ANY COMMITMENT OR REPRESENTATION. THE NAMES OF ACTUAL COMPANIES AND PRODUCTS MENTIONED HEREIN MAY BE THE TRADEMARKS OF THEIR RESPECTIVE OWNERS.