IFS Aviation Maintenance

eLogbook Configuration Guide

ICAM 26.2



Document Revision: 26.2-1

Publication Date: November 02, 2026

Contents

1 eLogbook API overview 4

2 API scenarios 6

eLogbook API scenarios 6

Create/Update logbook entry 6

Request fault deferral reference approval 6

Complete pilot task 7

Create fault and close 7

Query aircraft 7

Query faults 7

Query aircraft maintenance release status 8

Query flight legs 8

Query flight leg measurement requirements 8

Query persons 9

Query pilot tasks 9

3 Configuring eLogbook integration with Maintenix 10

Licensing 10

Permissions and authorization 10

Giving eLogbook integration permissions to a user role 11

Assigning the eLogbook integration role to a user 12

Outbound event framework 12

Changing the default password of the Maintenix mxextension user 12

Enabling the PubSub framework 13

Subscribing for outbound events 13

Activating outbound events 14

Troubleshooting eLogbook configuration 18

4 Create/Update logbook entry - overview 19

5 Request fault deferral reference approval - overview 20

Set up default deferral reference 20

6 Complete pilot task - overview 22

7 Create fault and close - overview..... 23

8 eLogbook API.....24
OpenAPI specification.....24
Message samples.....24
Create/Update logbook entry.....24
Request fault deferral reference approval.....26
Complete pilot task.....27
Create fault and close.....28
Query aircraft.....29
Query faults.....30
Query aircraft maintenance release status.....31
Query flight legs.....31
Query flight leg measurement requirements.....32
Query persons.....33
Query pilot tasks.....33

9 Errors and warnings.....34
Message response error codes.....34

1

eLogbook API overview

Electronic logbook applications enable seamless, mobile communication of faults and other logbook-related information between the flight crew and the ground maintenance organization who work in unison to return an aircraft to service on schedule for departure. The use of electronic logbook applications returns aircraft to service faster and improves data accuracy by eliminating transcription errors.

Typically, electronic logbook applications run on a mobile device, and can be used by the flight crew both in the air and while on the ground to review and update aircraft maintenance status as follows:

Flight crews can:

- View upcoming planned flights.
- View recent fault history retrieved from Maintenix, including open faults, deferred faults, and recently closed faults.
- Raise new faults that are automatically synchronized with Maintenix in real time when in-flight connectivity is available. If in-flight connectivity is not available, logged faults are shared as soon as connectivity is accessible.
- Update fault details in OPEN status.
- See information entered by mechanics in Maintenix, such as aircraft releases, in near real time.
- Request fault deferral.
- Complete pilot tasks, which are simple inspection tasks.
- Create and close logbook faults.

Further, Line Maintenance personnel can:

- See new faults raised through the electronic logbook application directly in Maintenix.
- See pilot tasks in Maintenix.

Maintenance Operations Control (MOC) personnel can:

- See pending deferral requests in Maintenix.

This document will help you configure your system to use the Maintenix eLogbook API.

2

API scenarios

eLogbook API scenarios

This section describes the scenarios supported by the API.

The supported scenarios contain the following sections:

- **Goal**: The purpose of the scenario.
- **Preconditions**: The conditions that must be met before the scenario can be executed.
- **Messaging**: The description of the messaging required for the scenario.
- **Result**: The outcome of the successful execution of the scenario.

Create/Update logbook entry

Goal: Allow a pilot or cabin crew to create (and leave open) a logbook fault or update an open logbook fault.

Preconditions: A pilot or cabin crew has found a new logbook fault or wants to update an existing open logbook fault.

Messaging: Refer the OpenAPI specification for mandatory fields and other details of the payload. See *Create/Update logbook entry* for sample messages. See *Create/Update logbook entry - overview* for more information on this scenario.

Result: A new logbook fault is created in a commit work package or an existing open logbook fault is updated, in Maintenix.

Request fault deferral reference approval

Goal: Allow pilots, flight crew, and MOC personnel to work together to identify deferrals for open logbook faults.

Preconditions: An open logbook fault exists in a work package.

Messaging: Refer the OpenAPI specification for mandatory fields and other details of the payload. See *Request fault deferral reference approval* for sample messages. See *Request fault deferral reference approval - overview* for more information on this scenario.

Result: An open logbook fault has a pending deferral reference if the reference requires MOC approval.

Complete pilot task

Goal: Allow a pilot to complete a simple executable requirement.

Preconditions: An active, simple, executable requirement task exists that meets the conditions given in *Complete pilot task - overview*.

Messaging: Refer the OpenAPI specification for mandatory fields and other details of the payload. See *Complete pilot task* for sample messages. See *Complete pilot task - overview* for more information on this scenario.

Result: The task is packaged and completed separately, and informative content is included in the history notes.

Create fault and close

Goal: Allow a pilot or cabin crew to create and close a logbook fault.

Preconditions: A pilot or cabin crew has found a simple fault and dealt with it immediately.

Messaging: Refer the OpenAPI specification for details of the response payload. See *Create fault and close* for sample messages. See *Create fault and close - overview* for more information on this scenario.

Result: A fault is created and assigned to a new work package, and both the fault and the work package are completed with the associated corrective action.

Query aircraft

Goal: Allow a pilot or cabin crew to retrieve a list of all aircraft managed in Maintenix.

Preconditions: None.

Messaging: Refer the OpenAPI specification for details of the response payload. See *Query aircraft* for sample messages.

Result: A list of all aircraft managed in Maintenix is retrieved.

Query faults

Goal: Allow a pilot or cabin crew to retrieve a list of logbook faults by aircraft and date.

Preconditions: At least one aircraft exists in Maintenix.

Messaging: Refer the OpenAPI specification for details of the response payload. See *Query faults* for sample messages.

Result: A list of open and deferred logbook faults is retrieved against the specified aircraft. If closedAfterDate is provided, the list of faults also includes those faults closed since the specified date.

Query aircraft maintenance release status

Goal: Allow a pilot or cabin crew to retrieve maintenance release status for the specified aircraft.

Preconditions: The aircraft exists in Maintenix.

Messaging: Refer the OpenAPI specification for details of the response payload. See *Query aircraft maintenance release status* for sample messages.

Result: Maintenance release status information is retrieved for the aircraft. The status includes a list of OPEN and DEFER faults against that aircraft. If closedAfterDate is provided, the list of faults closed since the specified date is also included.

Query flight legs

Goal: Allow a pilot or cabin crew to retrieve a list of planned flights associated with the specified aircraft, in between the given From Departure date and the To Departure date with the status "PLAN."

Preconditions: The aircraft has associated planned or delayed flights.

Messaging: Refer the OpenAPI specification for details of the response payload. See *Query flight legs* for sample messages.

Result: A list of planned and delayed flights associated with the specified aircraft is retrieved.

Query flight leg measurement requirements

Goal: Allow a pilot or cabin crew to retrieve a list of flight leg measurement requirements for a specified aircraft.

Preconditions: The aircraft has at least one associated flight leg.

Messaging: Refer the OpenAPI specification for details of the response payload. See *Query flight leg measurement requirements* for sample messages.

Result: A list of flight leg measurement requirements is retrieved for the specified aircraft.

Query persons

Goal: Allow a pilot or cabin crew to retrieve basic identity information about individuals and system users in Maintenix.

Preconditions: At least one user exists.

Messaging: Refer the OpenAPI specification for details of the response payload. See *Query persons* for sample messages.

Result: A list of all Maintenix users is retrieved.

Query pilot tasks

Goal: Allow a pilot or cabin crew to retrieve a list of active pilot tasks by aircraft.

Preconditions: At least one active pilot task exists.

Messaging: Refer the OpenAPI specification for details of the response payload. See *Query pilot tasks* for sample messages.

Result: A list of active pilot tasks is retrieved against the specified aircraft.

3

Configuring eLogbook integration with Maintenix

Configuring Maintenix to interact with the eLogbook application requires the following: verifying eLogbook licensing, authorization, subscriptions and roles; enabling the PubSub framework; and subscribing for outbound events and activating them.

Licensing

To use the Maintenix eLogbook API, your organization must hold a valid license for it.

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

Permissions and authorization

The eLogbook role has all required permissions enabled to utilize the eLogbook functionality.

Maintenix automatically creates a default user called `elbintegration` with the ELBINTEG role, which gives the user the permissions to authenticate and authorize inbound messages from eLogbook to Maintenix. Any other user created for this purpose should be assigned the ELBINTEG role.

The ELBINTEG role is assigned the following permission set, which includes all the necessary permissions to receive eLogbook inbound messages:

- EXT_ELOGBOOK_INBOUND

Inbound requests

The individual permissions associated with the EXT_ELOGBOOK_INBOUND permission set are listed below.

API_AIRCRAFT_EVENT_PARM

Scenario name: Get Aircraft

API_FAULT_EVENT_PARM

Scenario name: Get Faults

API_FLIGHT_EVENT_PARM

Scenario name: Get Flight Legs

API_FLIGHT_LEG_MEASUREMENT_REQUIREMENT_EVENT_PARM

Scenario name: Get Flight Measurement Requirement by Aircraft

API_MAINTENANCE_RELEASE_STATUS_EVENT_PARM

Scenario name: Get Maintenance Release Status by Aircraft

API_PERSON_EVENT_PARM

Scenario name: Get People

API_PILOT_TASK_EVENT_PARM

Scenario name: Get Pilot Tasks

API_LOGBOOK_ENTRY_MODIFIED_EVENT_PARM

Scenario name: Logbook Entry Modified

API_ELOGBOOK_REQUEST_DEFERRAL_REFERENCE_EVENT_PARM

Scenario name: Request Fault Deferral Reference Approval

API_ELOGBOOK_COMPLETE_PILOT_TASK_EVENT_PARM

Scenario name: Complete Pilot Task

API_CREATE_FAULT_AND_CLOSE_EVENT_PARM

Scenario name: Create Fault and Close

Giving eLogbook integration permissions to a user role

Grant users with a specific role the permission to send eLogbook messages.

Steps

1. From your **Admin To Do List**, click **Role Search**.
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 **eLogbook API** check box.

6. Click **OK**.

Assigning the eLogbook integration role to a user

Grant specific users the permission to send eLogbook messages. If an integration user is used to interface with the eLogbook API, it is sufficient for this role to be assigned to only that user. Depending on the application security configuration, it isn't necessary for eLogbook permissions to be assigned to every user of the application.

Steps

1. From your **Admin To Do List**, click **Role Search**.
2. Enter ELBINTEG 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**. Otherwise, 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**.

Outbound event framework

Changing the default password of the Maintenix mxextension user

The mxextensions user is a generic user used for internal communications within Maintenix.

Different components rely on the mxextensions user for various functions. For example, eLogbook events rely on the user for calling internal APIs.

The credentials for this user are specified in the `mx-servers.properties` file.

```
maintenix.extensions.username=mxextensions
maintenix.extensions.password=<password>
```

The username field specifies the name of this user. The password field must be filled in before you can enable any eLogbook events; otherwise, the internal event framework will not function.

Maintenix database authentication

When the Maintenix database is used as the authentication store (user IDs and passwords are stored in the Maintenix database), the password of mxextensions is stored in the UTL_USER table of the Maintenix database. The mxextensions user's password can be changed directly in the database by updating the value of the PASSWORD column in the UTL_USER table. Alternatively, a system administrator can login to the WebLogic Administration console and update it there.

LDAP authentication

If your organization uses an LDAP Directory as the authentication store for Maintenix, the password of mxextensions is stored in the LDAP Directory, and the mxextensions password should be updated there as well.

Enabling the PubSub framework

PubSub is the Maintenix publish-subscribe message delivery framework. The PubSub framework operates as a message broker and facilitates communication between the Maintenix eLogbook API and the eLogbook application. For steps on enabling the PubSub framework, see the *Maintenix PubSub User Guide*.

Subscribing for outbound events

To receive notification of Maintenix logbook fault and aircraft operational status updates from the eLogbook API, the eLogbook application must be configured as a subscriber to the message channels for outbound events by updating the `pubsub-subscriptions.properties` file.

> **Note** A subscription can be added for each event through the JMS or HTTP notification models.

For steps on how to set up the subscriptions, see the *Maintenix PubSub User Guide*.

eLogbook outbound events with corresponding PubSub channels:

Outbound event	PubSub channel
Aircraft Created Event	AIRCRAFT_CREATED
Aircraft Entered Maintenance	AIRCRAFT_ENTERED_MAINTENANCE
Aircraft Exited Maintenance	AIRCRAFT_EXITED_MAINTENANCE
Aircraft Locked Event	AIRCRAFT_LOCKED
Aircraft Registration Code Changed Event	AIRCRAFT_REGISTRATION_CODE_CHANGED

Outbound event	PubSub channel
Aircraft Unlocked Event	AIRCRAFT_UNLOCKED
Fault Modified Event	FAULT_MODIFIED
Flight Measurement Requirement Modified Event	FLIGHT_MEASUREMENT_REQUIREMENT_MODIFIED
Person Modified Event	PERSON_MODIFIED
Flight Leg Modified Event	FLIGHT_LEG_MODIFIED
Pilot Task Modified Event	PILOT_TASK_MODIFIED

Activating outbound events

You must activate the outbound events for eLogbook by enabling the applicable database triggers.

Steps

In your database, run the following SQL statement for the outbound events using the following triggers and class names:

```
 -- Fill in the class name for the event with the trigger class
UPDATE utl_trigger SET active_bool = 1 WHERE class_name = '<insert
 class_name here>'
;

commit;
```

MX_IN_CREATE

Trigger name: Aircraft Creation

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.engineering.event.aircraftcreated.Aircraft
CreatedEvent
```

MX_IN_OPERSTATUS

Trigger name: Aircraft Entered Maintenance (aircraft operator status changed to INM)

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.engineering.event.aircraftenteredmaintenan
ce.AircraftEnteredMaintenanceEvent
```

Trigger name: Aircraft Exited Maintenance (aircraft operator status changed from INM to another status)

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.engineering.event.aircraftexitedmaintenanc
e.AircraftExitedMaintenanceEvent
```

MX_IN_LOCKED

Trigger name: Aircraft Locked

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.engineering.event.aircraftlocked.AircraftL
ockedEvent
```

MX_AC_REGCD

Trigger name: Aircraft Code Changed

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.engineering.event.aircraftregistrationcode
changed.AircraftRegCodeChangedEvent
```

MX_IN_UNLOCKED

Trigger name: Aircraft Unlocked

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.engineering.event.aircraftunlocked.Aircraf
tUnlockedEvent
```

MX_EDIT_FAULT (fault details updated)

Trigger name: Fault Modify

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.faultmodified.FaultModifi
edEvent
```

MX_CF_CREATE (fault raised)

Trigger name: Fault Create

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.faultmodified.FaultModifi
```

```
edEvent
```

MX_CF_DEFER (deferred fault)

Trigger name: Fault Defer

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.faultmodified.FaultModifi
edEvent
```

MX_CF_CERTIFY (certified fault)

Trigger name: Fault Certify

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.faultmodified.FaultModifi
edEvent
```

MX_CF_NFF (marked as no fault found)

Trigger name: Fault NFF (No Fault Found)

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.faultmodified.FaultModifi
edEvent
```

MX_CF_ERROR (fault marked as error)

Trigger name: Mark Fault as Error

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.faultmodified.FaultModifi
edEvent
```

MX_CF_SET_OP_RESTRICTION

Trigger name: Set Operational Restriction

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.faultmodified.FaultModifi
edEvent
```

MX_FLIGHT_MEASUREMENT_REQUIREMENT_MODIFIED

Trigger name: Flight Measurement Requirement Modified

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.engineering.event.flightlegmeasurementrequ
irementmodified.FlightLegMeasurementRequirementModifiedEvent
```

MX_PERSON_MODIFIED

Trigger name: Person Modified

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.hr.event.personmodified.PersonModifiedEve
nt
```

MX_FL_MODIFIED

Trigger name: Flight Modified

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.flightops.event.flightlegmodified.FlightLe
gModifiedEvent
```

MX_TS_CREATE

Trigger name: Create Pilot Task

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.pilottaskmodified.PilotTa
skModifiedEvent
```

MX_TS_CANCEL

Trigger name: Cancel Pilot Task

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.pilottaskmodified.PilotTa
skModifiedEvent
```

MX_TS_COMPLETE

Trigger name: Complete Pilot Task

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.pilottaskmodified.PilotTa
skModifiedEvent
```

MX_TS_TERMINATE

Trigger name: Terminate Pilot Task

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.pilottaskmodified.PilotTa
skModifiedEvent
```

MX_TS_ERROR

Trigger name: Mark Pilot Task as an Error

Trigger class_name:

```
com.mxi.mx.amext.mxapiext.production.event.pilottaskmodified.PilotTa
skModifiedEvent
```

Troubleshooting eLogbook configuration

See the Troubleshooting PubSub section of the *Maintenix PubSub User Guide*.

4

Create/Update logbook entry - overview

The "create (and leave open) or update logbook fault" functionality is the backbone of the eLogbook API. It facilitates the flow of data from the air crew to maintenance to identify any issues found on the aircraft. This scenario is triggered when a pilot or cabin crew submits a new logbook fault in the electronic logbook application. When an OPEN fault is raised in Maintenix for a new logbook entry, if the logbook entry is modified and there exists a corresponding fault in Maintenix, the fault details are updated to match the logbook entry.

The POST request for this scenario is a synchronous message to invoke the Maintenix "raise fault" operation, and the successful response contains a representation of the newly created fault.

Note When a logbook fault is created by a Logbook Entry Modified message, the fault is assigned to an existing IN WORK work package. If there's no IN WORK work package, the fault is assigned to its own newly created work package in a state of COMMIT. The fault assignment is not changed when a logbook fault is updated by a Logbook Entry Modified message.

5

Request fault deferral reference approval - overview

A fault deferral request message provides the ability for a user in an external application to request deferral reference approval of an existing logbook fault.

Maintenix uses the deferral reference name to look up the deferral reference for the aircraft assembly and carrier. In the event the deferral reference provided is invalid or cannot be applied to the specified fault, a default deferral reference is used.

Note The messages must be generated and pushed in the same sequence in which the actions were performed in the external eLogbook application, to ensure desired behavior.

Note The value for `referenceName` in the message response corresponds to the actual value used, and not necessarily the value in the request. See sample messages for more information.

Set up default deferral reference

Setting up default deferral references for each of the aircraft assemblies in Maintenix is important for smooth integration workflow. Ensure to do the following during setup:

1. Go to the **Deferral Reference Search** To Do list button.
2. Click the **Create New** hyperlink to create a new Deferral Reference, or search and open an existing one to be used as the default.
3. Select all operators that will be using this deferral reference as a default.
4. Save the changes.
5. Go to the database.

Note Maintenix supports the configuration of default deferral references through a boolean field in the `fail_defer_ref` table. The column is called `assmbl_default_bool` and has the following attributes:

Defaulted to 0 (false). 1 (true) indicates a default value.

Boolean is editable only via a database update.

Only one default row is allowed per assembly. Any API request for deferral reference approval for the indicated aircraft may fail if more than one default is configured.

6. Update the `fail_defer_ref` table for the deferral reference above and set `assmbl_default_bool` = 1.
7. Commit the changes.

6

Complete pilot task - overview

The eLogbook API enables the flight crew to use the electronic logbook application to complete pilot tasks, which are simple inspection tasks. Tasks completed through this API are packaged and completed in separate work packages to avoid conflict with ongoing maintenance work packages. The tasks need to fulfill the following conditions:

- Executable REQ (i.e.: no job cards) not in a BLOCK
- No part requirements
- No panels
- No measurements
- No steps
- No tools
- No labor rows with e-signature
- The task is ACTV

If any of the above conditions are not met, the API does not complete the task, and an error message is returned. Therefore, a proper data setup needs to be ensured for the desired functioning of the API.

The API includes informative content by adding a user history note to the task history. This note logs the "recorded by" user from the authenticated request credentials of the user (often an integration user), with a message that indicates the value of the username provided in the metadata portion of the request.

Note To enable this functionality, the configuration parameter ELOGBOOK_PILOT_TASK_SUBCLASSES in `utl_config_parm` must be updated with a comma separated list of `task_subclass_cd` from `ref_task_subclass`.

7

Create fault and close - overview

The "create fault and close" operation creates and immediately closes a new logbook fault. This is useful when the issue identified in the fault is addressed immediately, and enables the eLogbook user to provide details of the corrective action describing what was done to fix the fault.

Note The "create fault and close" operation is used for raising faults in real-time. This operation is not used for future faults.

8

eLogbook API

OpenAPI specification

You can find the eLogbook OpenAPI specification at the URL

`http://{{eLogbookHost}}/mxapiext/elogbook-openapi.json`

on your server.

Message samples

This section contains sample requests/responses.

Create/Update logbook entry

This message is used to create (and leave open) a logbook fault or update an open logbook fault.

POST `http://{{eLogbookHost}}/mxapiext/flight-ops/logbook-entry-modified`

Request

```
{
    "metadata": {
        "timestamp": "2024-03-01T07:48:11.169Z",
        "system": "Maintenix",
        "user": "mxintegration"
    },
    "logbookReference": "224400117700",
    "name": "Center, upper DU INOP",
    "description": "Center, upper DU INOP in all switch positions or
 configurations.",
    "logbookType": "TECH",
    "failedSystem": "05-50-00",
    "source": "PILOT",
```

```
    "foundOnAircraft": "D746F7925BC74C6B91ED720E491FACC9",
    "foundDuringFlightLeg": "B901FCF2AB1611EF841A5414F36A85BF",
    "foundDuringFlightPhase": "APR",
    "foundOnDate": "2024-03-01T07:48:11.169Z",
    "foundByPerson": "100",
    "faultCode": "114",
    "resultingEvents": ["ABT","ATB"]
}
```

Success response

```
{
    "faultId": "286C4CEB92451D47E063030011AC52B6",
    "barcode": "T40S0001DR2",
    "logbookReference": "224400117700",
    "name": "Center, upper DU INOP",
    "description": "Center, upper DU INOP in all switch positions or
 configurations.",
    "status": "OPEN",
    "logbookType": "TECH",
    "source": "PILOT",
    "foundOnAircraft": "D746F7925BC74C6B91ED720E491FACC9",
    "foundDuringFlightLeg": "B901FCF2AB1611EF841A5414F36A85BF",
    "foundDuringFlightPhase": "APR",
    "foundOnDate": "2024-03-01T07:48:11.171Z",
    "foundByPerson": "100",
    "failedSystem": "05-50-00",
    "faultCode": "114",
    "deferralReference": null,
    "deferralClass": null,
    "deferralAuthorization": null,
    "schedulingRules": [],
    "correctiveAction": null,
    "completedByPerson": null,
    "completedOnDate": "2024-03-01T07:48:11.171Z",
    "operationalRestriction": null,
    "resultingEvents": [
        "ABT",
        "ATB"
    ],
    "metadata": {
        "timestamp": "2024-03-01T07:48:11.171Z",
        "system": "Maintenix",
        "user": "mxintegration"
    }
}
```

Error response

```
{
  "errorMessage": "aircraft cannot be empty or null",
  "metadata": {
```

```
    "timestamp": "2024-03-01T07:48:11.173Z",
    "system": "Maintenix",
    "user": "mxintegration"
  }
}
```

Request fault deferral reference approval

This message is used to identify deferrals for open logbook faults.

POST `http://{{eLogbookHost}}/mxapiext/flight-ops/request-deferral-reference-approval`

Request

```
{
  "logbookReference": "string",
  "aircraft": "EB83F39AD5F70A3FE053020011ACBBCA",
  "referenceName": "string",
  "reason": "string",
  "notes": "string",
  "metadata": {
    "timestamp": "2024-03-01T08:13:09.681Z",
    "system": "Maintenix",
    "user": "mxintegration"
  }
}
```

Success response

```
{
  "logbookReference": "string",
  "aircraft": "EB83F39AD5F70A3FE053020011ACBBCA",
  "referenceName": "string",
  "reason": "string",
  "notes": "string",
  "metadata": {
    "timestamp": "2024-03-01T08:13:09.691Z",
    "system": "Maintenix",
    "user": "mxintegration"
  }
}
```

Error response

```
{
    "errorMessage": "Unable to find fault for logbook entry 123 on
 aircraft C0B453B0A471D1A6E0539406800A5A7E.",
    "metadata": {
        "timestamp": "2024-02-29T15:16:29Z",
```

```
        "system": "Maintenix",
        "user": "user5"
    }
}
```

Complete pilot task

This message is used to complete a simple executable requirement.

POST `http://{{eLogbookHost}}/mxapiext/flight-ops/complete-pilot-task`

Request

```
{
  "taskId": "EB83F39AD5F70A3FE053020011ACBBCA",
  "actualEndDate": "2024-03-01T08:06:38.465Z",
  "metadata": {
    "timestamp": "2024-03-01T08:06:38.465Z",
    "system": "Maintenix",
    "user": "mxintegration"
  }
}
```

Success response

```
{
  "taskId": "EB83F39AD5F70A3FE053020011ACBBCA",
  "actualEndDate": "2024-03-01T08:06:38.466Z",
  "metadata": {
    "timestamp": "2024-03-01T08:06:38.466Z",
    "system": "Maintenix",
    "user": "mxintegration"
  }
}
```

Error response

```
{
    "errorMessage": "The property
 completePilotTask.arg0.completePilotTask.taskId with value [123] is
 invalid. Reason: must match \"^[0-9A-F]{32}$\".",
    "metadata": {
        "timestamp": "2024-02-29T15:20:36Z",
        "system": "Maintenix",
        "user": "user1"
    }
}
```

Create fault and close

This message is used to create and close a logbook fault.

POST `http://{{eLogbookHost}}/mxapiext/flight-ops/logbook-entry-create-and-close`

Request

```
{
    "metadata": {
        "timestamp": "2024-03-01T07:48:11.169Z",
        "system": "Maintenix",
        "user": "mxintegration"
    },
    "logbookReference": "224400117700",
    "name": "Center, upper DU INOP",
    "description": "Center, upper DU INOP in all switch positions or
 configurations.",
    "logbookType": "TECH",
    "failedSystem": "05-50-00",
    "source": "PILOT",
    "foundOnAircraft": "D746F7925BC74C6B91ED720E491FACC9",
    "foundDuringFlightLeg": "B901FCF2AB1611EF841A5414F36A85BF",
    "foundDuringFlightPhase": "APR",
    "foundOnDate": "2024-03-01T07:48:11.169Z",
    "foundByPerson": "100",
    "faultCode": "114",
    "correctiveAction": "Inspected, Replaced",
    "resultingEvents": ["ABT","ATB"]
}
```

Success response

```
{
    "faultId": "286C4CEB92451D47E063030011AC52B6",
    "barcode": "T40S0001DR2",
    "logbookReference": "224400117700",
    "name": "Center, upper DU INOP",
    "description": "Center, upper DU INOP in all switch positions or
 configurations.",
    "status": "OPEN",
    "logbookType": "TECH",
    "source": "PILOT",
    "foundOnAircraft": "D746F7925BC74C6B91ED720E491FACC9",
    "foundDuringFlightLeg": "B901FCF2AB1611EF841A5414F36A85BF",
    "foundDuringFlightPhase": "APR",
    "foundOnDate": "2024-03-01T07:48:11.171Z",
    "foundByPerson": "100",
    "failedSystem": "05-50-00",
    "faultCode": "114",
    "deferralReference": null,
    "deferralClass": null,
```

```
    "deferralAuthorization": null,
    "schedulingRules": [],
    "correctiveAction": "Inspected, Replaced",
    "completedByPerson": null,
    "completedOnDate": "2024-03-01T07:48:11.171Z",
    "operationalRestriction": null,
    "resultingEvents": [
        "ABT",
        "ATB"
    ],
    "metadata": {
        "timestamp": "2024-03-01T07:48:11.171Z",
        "system": "Maintenix",
        "user": "mxintegration"
    }
}
```


Error response


```
{
  "errorMessage": "aircraft cannot be empty or null",
  "metadata": {
    "timestamp": "2024-03-01T08:19:41.913Z",
    "system": "Maintenix",
    "user": "mxintegration"
  }
}
```


Query aircraft

This message is used to retrieve a list of all aircraft managed in Maintenix.

GET request

```
http://{{eLogbookHost}}/mxapiext/engineering/aircraft
```

Response


```
[
  {
    "id": "873A967DB4B44404AB0F13ECEB3C8B68",
    "registrationCode": "N123AB",
    "isLocked": true
  }
]
```

Query faults

This message is used to retrieve a list of logbook faults by aircraft and date.

GET request

```
http://{{eLogbookHost}}/mxapiext/production/fault?
ClosedAfterDate=2023-10-31&aircraft=873A967DB4B44404AB0F13ECEB3C8B68
```

Response


```
[
  {
    "faultId": "873A967DB4B44404AB0F13ECEB3C8B68",
    "barcode": "I0004VMXABCD",
    "logbookReference": "AC430-23362-001",
    "name": "Center, upper DU INOP",
    "description": "Center, upper DU INOP in all switch positions or
 configurations.",
    "status": "DEFER",
    "logbookType": "TECHCAB",
    "source": "MECH",
    "foundOnAircraft": "873A967DB4B44404AB0F13ECEB3C8B68",
    "foundDuringFlightLeg": "873A967DB4B44404AB0F13ECEB3C8B68",
    "foundDuringFlightPhase": "TXI",
    "foundOnDate": "2024-03-01T08:36:13.029Z",
    "foundByPerson": "user1",
    "failedSystem": "05-50-00",
    "faultCode": "114",
    "deferralReference": "MEL 21-65-2",
    "deferralClass": "MEL C",
    "deferralAuthorization": "102467",
    "schedulingRules": [
      {
        "parameter": "HOURS",
        "start": "514.5",
        "due": "580.0",
        "extension": "100.0"
      }
    ],
    "correctiveAction": "Ok to continue per MEL",
    "completedByPerson": "user1",
    "completedOnDate": "2024-03-01T08:36:13.029Z",
    "operationalRestriction": "AIRSPEED RESTRICTED. INCREASED FUEL
 BURN ON EXTEND FLTS",
    "resultingEvents": [
        "ABT",
        "ATB"
    ]
  }
]
```

Query aircraft maintenance release status

This message is used to retrieve maintenance release status for the specified aircraft.

GET request

```
http://{{eLogbookHost}}/mxapiext/engineering/maintenance-release-statu
s?ClosedAfterDate=2023-10-31&aircraft=873A967DB4B44404AB0F13ECEB3C8B68
```

Response

```
{
  "aircraft": "873A967DB4B44404AB0F13ECEB3C8B68",
  "isInMaintenance": true,
  "crs": {
    "etopsReady": true,
    "certifier": "user1",
    "timeStamp": "2024-03-01T08:52:08.964Z",
    "releaseNumber": "WO - 123456",
    "releaseRemarks": "Release remark for ABC aircraft."
  },
  "faults": [
    {
      "faultId": "873A967DB4B44404AB0F13ECEB3C8B68",
      "barcode": "I0004VMXABCD",
      "status": "OPEN",
      "deferralReference": "DeferralReference01"
    }
  ]
}
```

Query flight legs

This message is used to retrieve a list of planned flights associated with the specified aircraft, in between the given From Departure date and the To Departure date with the status "PLAN."

GET request

```
http://{{eLogbookHost}}/mxapiext/flight-ops/flight-leg?aircraft=
873A967DB4B44404AB0F13ECEB3C8B68
```

Response

```
[
  {
    "id": "873A967DB4B44404AB0F13ECEB3C8B68",
    "aircraft": "873A967DB4B44404AB0F13ECEB3C8B68",
    "aircraftRegistrationCode": "N123AB",
    "flightNumber": "0568",
    "status": "PLAN",
```

```
    "departure": {
      "airport": "ATT",
      "scheduledDate": "2024-03-01T09:00:15.997Z",
      "estimatedDate": "2024-03-01T09:00:15.997Z",
      "actualDate": "2024-03-01T09:00:15.997Z",
      "upDate": "2024-03-01T09:00:15.997Z",
      "gate": "A18"
    },
    "arrival": {
      "airport": "ABQ",
      "scheduledDate": "2024-03-01T09:00:15.997Z",
      "estimatedDate": "2024-03-01T09:00:15.997Z",
      "actualDate": "2024-03-01T09:00:15.997Z",
      "downDate": "2024-03-01T09:00:15.997Z",
      "gate": "B15"
    },
    "reason": "OD",
    "operationalType": "FERRY",
    "description": "Ferry flight to return equipment."
  }
]
```


Query flight leg measurement requirements

This message is used to retrieve a list of flight leg measurement requirements for a specified aircraft.

GET request

```
http://{{eLogbookHost}}/mxapiext/engineering/flight-leg-measurement-
requirement?aircraft=873A967DB4B44404AB0F13ECEB3C8B68
```

Response


```
[
  {
    "aircraft": "873A967DB4B44404AB0F13ECEB3C8B68",
    "measurements": [
      {
        "dataTypeCode": "Aircraft weight",
        "engineeringUnitCode": "Kg"
      }
    ]
  }
]
```

Query persons

This message is used to retrieve basic identity information about individuals and system users in Maintenix.

GET request

```
http://{{eLogbookHost}}/mxapiext/hr/person
```

Response

```
[
  {
    "id": "100",
    "firstName": "Philip",
    "middleName": "Benedict",
    "lastName": "Thomas"
  }
]
```

Query pilot tasks

This message is used to retrieve a list of active pilot tasks by aircraft.

GET request

```
http://{{eLogbookHost}}/mxapiext/production/pilot-task?aircraft=
873A967DB4B44404AB0F13ECEB3C8B68
```

Response

```
[
  {
    "aircraft": "873A967DB4B44404AB0F13ECEB3C8B68",
    "taskId": "ABCA967DB4B44404AB0F13ECEB3C8B68",
    "barcode": "T40S0001DQM",
    "taskCode": "Visual Inspection of Nose Landing Gear Strut",
    "name": "Visual Inspection of Nose Landing Gear Strut for Leaks",
    "status": "ACTV",
    "description": "Maintenance Task to be completed by the pilot",
    "dueDate": "2024-03-01T10:01:02.903Z"
  }
]
```

9

Errors and warnings

Message response error codes

The following response codes are returned in response to eLogbook API requests.

Message response error codes

HTTP Status Code	Scenario
200	A success. The body will contain the payload outlined in the message samples sections.
400	Bad request. The request received from the consumer cannot be processed because the request data is not valid or the request target is not in a valid state to apply the requested change. The consumer may amend the request and try again.
401	The request failed an authentication check (the user does not exist within Maintenix).
403	The user is valid but isn't authorized to access one or more of the APIs, or the license is not enabled for the eLogbook feature.
404	The URL is invalid. This may occur if the path is invalid, or if one of the query parameters failed schema validation (e.g., the UUID is not 32 hex characters).

HTTP Status Code	Scenario
500	The request cannot be processed because the system has not been properly configured or an unexpected error has occurred. Consult the Maintenix log file for further details.

Sample error message

```
{
    "errorMessage": "Task 06BD2F454B055393E06343401BAC2B04 is not a
 known task id.",
    "metadata": {
        "timestamp": "2023-11-28T20:56:09Z",
        "system": "Maintenix",
        "user": "User1"
    }
}
```

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.