IFS Aviation Maintenance

ARC API Guide

ICAM 26.2



Document Revision: 26.2-1

Publication Date: November 02, 2026

Contents

1 Introduction....4
What's New in This Release....4
The CAMO ARC API....4

2 API usage....6
Overview....6
Upload endpoint....7
Upload status endpoint....8
Upload result endpoint....9

3 ARC API configuration....11
API permissions....11
ARC API health check....11

4 API endpoint details....12
API conventions....12
Content types....13
Error response format....13
Authorization....13
Operation attribution....13
Correlation id....14
OpenAPI specification....14
Sample request/response....14
Upload request....14
Sample request/response....14
Request details....17
Root elements....18
Asset element....18
Aircraft details....19
Current configuration element....21
Inventory details element....24
Inventory usages element....26
Completed tasks element....28
Open faults element....30

Response details....36
Error responses....36
Upload status....36
Sample request/response....36
Response details....37
Error responses....38
Upload result....38
Sample request/response....38
Response details....40
Asset log entry details....41
Error responses....42
ARC health check....42
Sample request/response....43
Response details....43
ARC error responses....43
Sample error response....43
Response details....44
Standard error codes....45

1 Introduction

What's New in This Release

- Support for loose tracked inventory (TRK) as the root asset in ARC uploads.

The CAMO ARC API

The Asset Record Coordination (ARC) API can be used to load an asset's (aircraft, sub-assembly, or TRK inventory) technical records such as aircraft details, inventory details, current configuration, inventory usage information, open faults details, and completed tasks from an external system. The information can come from one or several external systems, with each system providing the information for one or several of the elements contained in the message. For example, one external system provides information about inventory and usage details and another system provides open fault details for each aircraft in the fleet. Maintenix uses this information to create and maintain aircraft records, and facilitate maintenance planning and execution, including Production Planning and Control (PP&C) and Long Range Planner (LRP).

The ARC API is not a bi-directional interface. An asset's technical records can be sent to Maintenix from an external system, but once updated, the records cannot be sent back to the external system. However, from the ARC Message Management page in Maintenix, you can manually export an aircraft's technical records to an XML file, which can then be imported to the external system. The ARC API supports technical record uploads where the root asset is an aircraft (ACFT), a loose assembly (ASSY), or a loose tracked inventory (TRK). All configuration, usage, task, and fault data in the request is scoped to the specified root asset and its sub-components.

For more information about asset management, technical records, and planning activities, refer to the *User Guide*. Functional details of the types of records that can be transferred using ARC and

the business logic of how ARC processes them can be found in the Asset Record Coordination section of the *User Guide*.

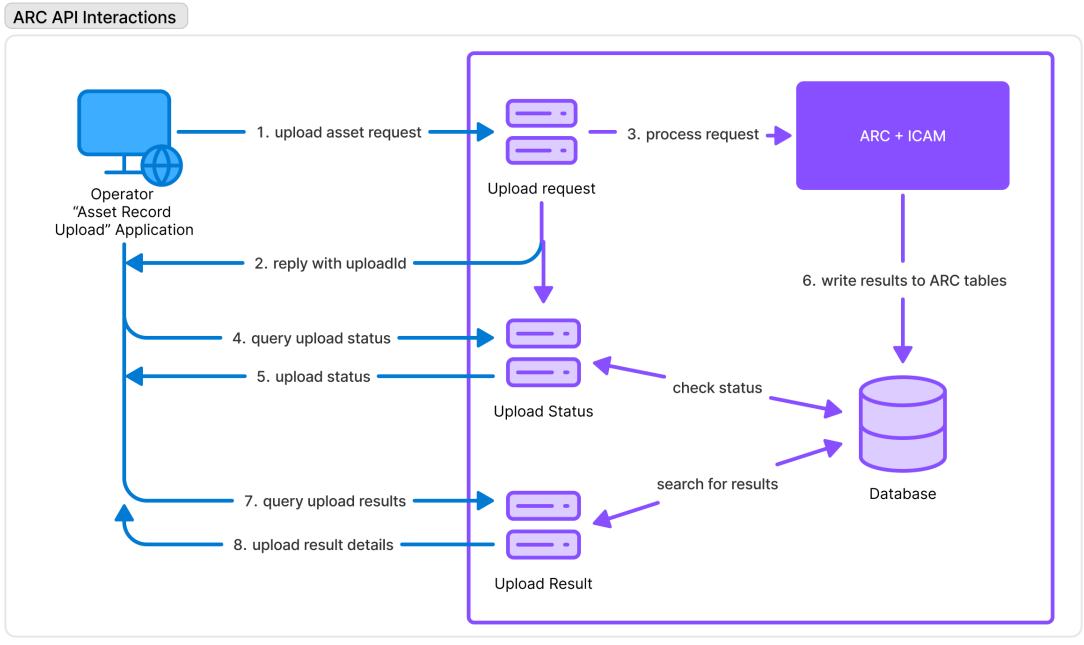
Using the ARC API also requires that ARC be configured correctly. Details about the user permissions, ARC-specific configuration parameters, and some general Maintenix configuration parameters that impact ARC processing are described in the Asset Record Coordination section of the *User Guide.*

2

API usage

Overview

This section describes how the ARC API can be orchestrated to transfer technical records related to an asset into CAMO and then view the results associated with the transfer.



An external application can use the Upload endpoint to initiate the transfer of asset records into CAMO. This endpoint will stage the asset details and respond immediately with an acknowledgment that includes the unique identifier associated with the upload request, and, if the staging is successful, will start processing the upload request in the background in parallel.

As the upload progresses, an external application can query the status of the upload using the upload id provided in the initial response.

Once the processing of the upload has completed, the external application can query the upload results to retrieve the upload's validation log entries: the list of errors and warnings associated with the records transfer.

Upload endpoint

Request

`POST https://<host:port>/arc/api/uploads`

The Upload endpoint supports the ability to initiate an upload of the following types of technical records for a single asset and its sub-components:

- Aircraft registration and operating details
- Sub-inventory configuration
- Inventory details
- Current usages
- Completed (last-done) tasks
- Open faults

See *Upload request* for data structure details. The API upload is functionally equivalent to the Upload capability of the ARC Message Management UI screen, and complete details about how upload requests are processed by ARC are documented in the Asset Record Coordination section of the *User Guide*. The OpenAPI specification for the API upload request also closely mirrors the `asset-record-coordination.xsd` used in ARC Message Management uploads, with the exception that the ARC API does not support uploading technical records for multiple assets in a single request—only one asset per invocation is allowed.

When ARC receives an asset upload request, it creates or updates the technical records for the asset and all sub-components included in the request. If a matching asset record is found in CAMO, the technical records are updated. If a matching record is not found in CAMO, a new asset is created along with the technical records for that asset.

The external application also has the option to associate the upload request with their own identifier using the "externalReference" property. If provided, this identifier will be included in all operations associated with the upload, including subsequent status requests and it will be visible in the ARC Message Management UI screen.

The Upload request supports two processing workflows: VALIDATE_ONLY and VALIDATE_AND_PROCESS.

- In the VALIDATE_ONLY workflow, the API parses and validates the request, but does not update records in Maintenix.

- In the VALIDATE_AND_PROCESS workflow, the API both validates the request and applies updates to Maintenix records.

A key limitation of the VALIDATE_ONLY workflow is that some validations in Maintenix depend on how data is actually stored and connected in the database. Because VALIDATE_ONLY does not update or arrange data in the database, certain errors or issues may not be detected until the VALIDATE_AND_PROCESS workflow is used. These deeper checks occur only when real updates are performed.

If no workflow is specified, the API defaults to VALIDATE_ONLY.

> **Note** Root asset type validation: ARC accepts upload requests only when the root asset identified by asset.id is an aircraft (ACFT), assembly (ASSY), or tracked inventory (TRK). If the referenced inventory is of any other class, the request fails validation and no configuration, usage, task, or fault data is processed.

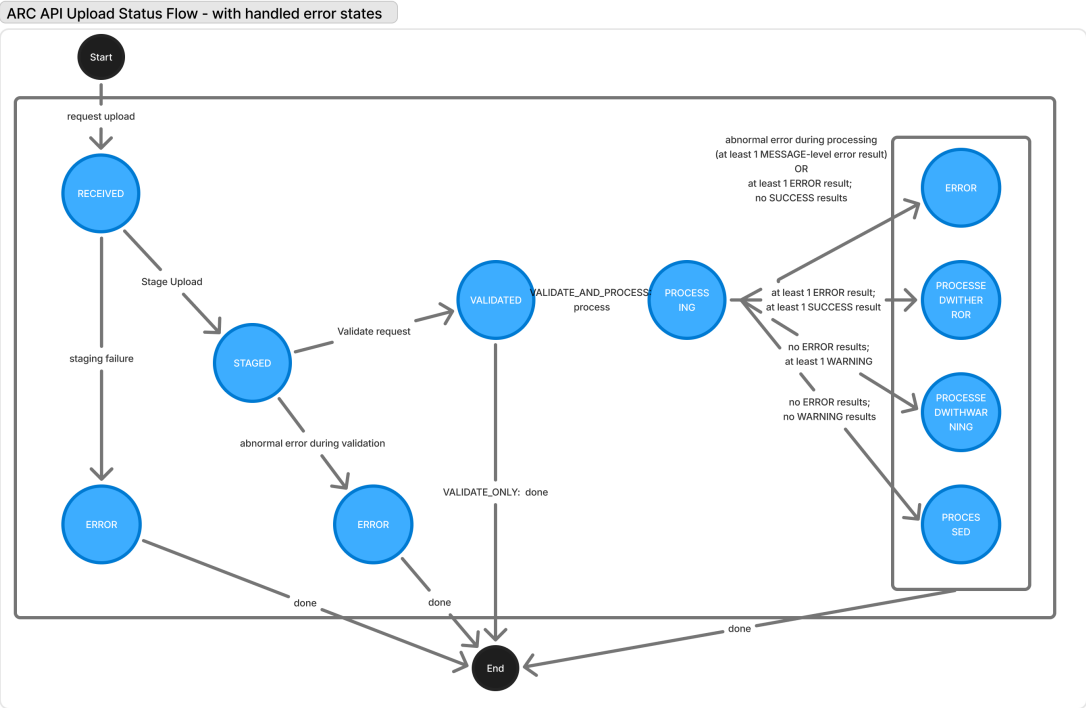
A detailed list of validation findings for each workflow is available in the Asset Record Coordination section of the *User Guide*.

Response

Upon successful parsing of the upload request, the API stages the asset details, assigns the uploadId to the request and responds with an HTTP status code `202 Accepted`. The response payload is the Upload Status entity (see *Upload status endpoint*), which includes the uploadId required in subsequent API interactions related to this upload.

Upload status endpoint

External applications can use the Upload Status endpoint to poll for the status of an ongoing or completed upload request. The following state diagram describes the status transitions of upload processing:



Request

`GET https://<host:port>/arc/api/uploads/status/<uploadId>`

The Upload Status endpoint supports only GET and requires that the specific uploadId be provided in the URL path.

This endpoint does not support the ability to query the status of uploads by any other filter.

Response

The Upload Status endpoint response indicates the current status of the upload, as well as including some details about the upload request. The completionDate property of the response indicates when the upload processing has finished: this value will be null until the upload completes.

The status 'VALIDATED' serves as the final state for upload requests processed with the VALIDATE_ONLY workflow. In contrast, for requests using the VALIDATE_AND_PROCESS workflow, 'VALIDATED' represents an intermediate state. To determine whether 'VALIDATED' indicates completion, review the workflow type in conjunction with the status, or confirm that the completion date has been populated.

Upload result endpoint

After the upload process is complete, the status will indicate whether the request was fully or partially processed. During processing, data elements may be modified to resolve conflicts,

defaulted if not provided, or otherwise altered to facilitate successful data import. All such modifications, along with validation and processing outcomes for each element of the request, are recorded in the validation log. The Upload Result endpoint provides access to this log.

For comprehensive information regarding the types of outcomes and modifications reported by ARC, see the Asset Record Coordination section of the *User Guide*.

Request

`GET https://<host:port>/arc/api/uploads/result/<uploadId>`

The Upload Result endpoint supports only GET and requires that the specific uploadId be provided in the URL path.

If the upload is still in progress, this endpoint will respond with a `404 Not Found`. Consumers should ensure that the upload has completed prior to invoking this request.

This endpoint does not support the ability to query the results of uploads by any other filter.

Response

The Upload Result response provides the overall outcome of the upload (see state diagram in *Upload status endpoint*) as well as the validation log.

3

ARC API configuration

API permissions

Access to the ARC API requires that the user specified in the Authentication header has the API_ARC permission, along with all related functional permissions. These permissions are detailed in the Permission Settings section under the Asset Record Coordination section of the *User Guide*.

ARC API health check

The ARC API provides a Health Check endpoint that enables Dev Ops and/or consumer applications to confirm that the ARC API web service is properly deployed and operational. This lightweight endpoint helps system administrators to confirm URL configuration of network infrastructure at the HTTP level and successful deployment of the ARC web service within the CAMO application. See *API endpoint details* for detailed access information.

Note Since this API endpoint does not provide business functionality, it is not necessary to grant functional business permissions to users invoking this endpoint.

Request

`GET https://<host:port>/arc/api/health`

The Health Check endpoint supports only GET and requires no path parameters.

Response

The successful health check response indicates that the ARC API is available to provide service.

4

API endpoint details

This section provides detailed information for each of the API endpoints. For each endpoint, the following information is provided:

- **Sample Request/Response Payloads:** An example of each request (where applicable) and response payload.
- **Payload Contents:** Each section includes one or more tables that outline the request/response content details.
  - **Usage Flags**: Identifies whether each element is mandatory or optional, and whether it has any specific or uniqueness constraints.
    - **M/O** indicates whether the element is mandatory or optional.
    - **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in CAMO.
    - **U** indicates that the element, if provided, must be unique within CAMO.
  - **Mapping Elements to the CAMO UI:** Identifies the corresponding field in the CAMO user interface for each element in the payload, enabling traceability between API data and UI components.
  - **Notes**: Elaborates on usage flags and constraints identified in the table. These notes are not intended to include full details of the element behavior or processing logic. For comprehensive information on solution behavior, see the Asset Record Coordination section of the *User Guide* and the ARC OpenAPI specification available in the product documentation.
- **Errors**: A list of errors that can be encountered when sending messages to the endpoint.

API conventions

The following conventions establish the foundational technical rules that apply to all ARC API endpoints. These guidelines ensure consistency in how requests and responses are structured,

how errors are communicated, and how authentication is handled across the API. Understanding these conventions will help developers and integrators interact reliably with any endpoint described in this guide.

Content types

All requests and responses in the ARC API use the media type `application/json`.

- Clients must set the `Content-Type` header to `application/json` in all requests with a body.
- If the `Accept` header is sent, it must allow for `application/json` in the response.
- The API will respond with `Content-Type: application/json` in all responses.

Error response format

All error responses follow a consistent, structured format using media type `application/problem+json`. Each error response includes the HTTP status code of the response as well as an application-specific error code to help identify the error's cause.

Full details of the format are documented in *ARC error responses*, along with common error codes that apply to all endpoints. Endpoint-specific error codes are documented as part of each endpoint.

Authorization

All endpoints require authentication via the HTTP Authorization header, and authorization to use each endpoint is granted based on the permissions granted to the user identified by the header.

- Access to the business logic endpoints of ARC (request upload, view results) requires that the user is granted the API_ARC permission.
- Retrieving the OpenAPI specification requires the API_VIEW_REQUEST permission.

Operation attribution

When processing an upload request, ARC invokes operations in CAMO that affect maintenance tracking and compliance history, such as attaching and detaching inventory, completing tasks, and raising and deferring faults. These operations are recorded in the history log of the relevant components and are attributed to the user whose credentials were provided in the Authorization header of the upload request.

Correlation id

The ARC API supports the use of the `X-Correlation-ID` HTTP header to facilitate end-to-end tracking of requests as they pass through the system. Clients can include a value for this header with each API request to help track and correlate message flow across different services and logs. If not specified, the ARC API will automatically generate a correlation ID and include it in the response.

OpenAPI specification

ARC API's OpenAPI specification provides a comprehensive reference for all endpoints, request and response schemas, validation rules, and processing behavior. It is available in the product documentation and at runtime from the ARC API.

Access to this endpoint is controlled by the API_VIEW_REQUEST permission.

Sample request/response

Request

`GET https://<host:port>/arc/api/openapi.json`

Success Response

`200 Ok`

The response payload is the OpenAPI specification for the ARC API, in JSON format.

Upload request

The asset record upload request message is sent by external systems to transfer/update asset information in CAMO to ensure the coordination of an asset's technical records across systems.

Sample request/response

Request

`POST https://<host:port>/arc/api/uploads`

```
{
    "workflow": "VALIDATE_AND_PROCESS",
    "externalReference": "ext-ref-1234",
    "asset": {
        "id": {
            "manufacturerCode": "ABI",
            "partNoOem": "A319/A320",
```

```
        "serialNumber": "33410"
    },
    "aircraftDetails": {
        "country": "CA",
        "registrationCode": "33410",
        "regulatoryBody": "FAA"
    },
    "currentConfiguration": [
        {
            "id": {
                "partGroupCode": "72-00-CFM6",
                "positionName": "1.LT"
            },
            "inventoryId": {
                "manufacturerCode": "CFM",
                "partNoOem": "CFM56-3C",
                "serialNumber": "SN000279"
            },
            "installation": {
                "date": "2025-10-10T00:00:00Z"
            }
        },
        {
            "id": {
                "partGroupCode": "70-00-02-12-4-165",
                "positionName": "1"
            },
            "inventoryId": {
                "manufacturerCode": "CFM",
                "partNoOem": "332A1024-2",
                "serialNumber": "SN000280"
            },
            "nhAssembly": {
                "manufacturerCode": "CFM",
                "partNoOem": "CFM56-3C",
                "serialNumber": "SN000279"
            }
        }
    ],
    "inventoryDetails": [
        {
            "inventoryId": {
                "manufacturerCode": "CFM",
                "partNoOem": "CFM56-3C",
                "serialNumber": "SN000279"
            },
            "originalEquipment": {
                "manufacturerCode": "CFM",
                "partNoOem": "CFM56-3C",
                "serialNumber": "SN000279"
            },
            "applicabilityCode": "200",
            "manufacturedDate": "2025-06-10T00:00:00Z",
            "receivedDate": "2025-06-11T00:00:00Z",
            "releaseNumber": "120"
```

```
            }
        ],
        "inventoryUsages": [
            {
                "inventoryId": {
                    "manufacturerCode": "CFM",
                    "partNoOem": "CFM56-3C",
                    "serialNumber": "SN000279"
                },
                "usages": [
                    {
                        "parameter": "B1",
                        "tsn": 28950.0,
                        "tso": 5460.0,
                        "tsi": 5120.0
                    },
                    {
                        "parameter": "B2",
                        "tsn": 46320.0,
                        "tso": 8740.0,
                        "tsi": 8200.0
                    }
                ]
            }
        ],
        "completedTasks": [
            {
                "inventoryId": {
                    "manufacturerCode": "CFM",
                    "partNoOem": "CFM56-3C",
                    "serialNumber": "SN000279"
                },
                "code": "Eng Task 1",
                "completionDate": "2025-05-16T00:30:00Z",
                "usagesAtCompletion": [
                    {
                        "parameter": "AOC",
                        "tsn": 28950.0
                    },
                    {
                        "parameter": "AOH",
                        "tsn": 46320.0
                    }
                ]
            }
        ],
        "openFaults": [
            {
                "inventoryId": {
                    "manufacturerCode": "CFM",
                    "partNoOem": "CFM56-3C",
                    "serialNumber": "SN000279"
                },
                "foundOnDate": "2025-11-03T21:28:00Z",
                "logbookReference": "Eng Fault 1",
```

```
                "name": "Eng Fault 1",
                "severity": "UNKNOWN",
                "source": "PILOT",
                "usagesWhenFound": [
                    {
                        "parameter": "B1",
                        "tsn": 1299
                    },
                    {
                        "parameter": "B2",
                        "tsn": 5343.2
                    }
                ]
            }
        ]
    }
}
```

Success response

`202 Accepted`:

```
{
    "uploadId": "19070017ADCF11F0B1E24851C5518011",
    "workflow": "VALIDATE_AND_PROCESS",
    "status": "STAGED",
    "receivedDate": "2025-10-20T16:09:03Z",
    "completedDate": null,
    "externalReference": "ext-ref-1234"
}
```

Request details

The ARC upload request contains a complex payload composed of several substantial sub-elements. To enhance clarity and maintain readability, this documentation breaks down the payload into subsections, each focusing on one major sub-element. These sections provide detailed descriptions to help developers understand the structure and purpose of each component within the overall request.

InventoryId

The inventoryId object is a composite identifier used throughout the ARC API to uniquely reference an asset or sub-component within CAMO. It consists of the following three elements:

- manufacturerCode
- partNoOem
- serialNumber

Within the upload request, it appears in multiple elements to specify which inventory item is being referenced and/or updated. The combination of these three properties is used to uniquely identify a tracked asset/component. It is used by ARC to match, create, or update inventory records in CAMO. If any inventoryId is incomplete (any element is not provided), the upload request will not be processed.

In the upload result, the validation log entries use the inventoryId to provide traceability between the API data and CAMO UI.

Root elements

The root of the ARC upload request consists of three properties that establish the context and control flow for the operation: workflow, externalReference, and asset. The workflow parameter guides ARC in determining the appropriate processing logic for the request. The externalReference is a consumer-defined identifier that ARC does not interpret, but which can be used to correlate the request across external systems. The asset object serves as the container element for the detailed technical records to be transferred.

Tracking information

Property name	Usage flags	UI mapping	Notes
workflow	O	n/a	Valid values: VALIDATE_ONLY (default) VALIDATE_AND_PROCESS
externalReference	O	n/a	
asset	M	Inventory Details	See section *Asset element*.

Asset element

The `asset` element serves as the root container for all of the technical records for the asset. Its purpose is to encapsulate structured data related to the asset and its sub-components. The technical records may include the asset's current configuration, completed maintenance tasks, usage data, descriptive details, and open faults.

To determine whether an asset is being newly inducted or an existing asset is being updated, ARC compares the `id` element in the request against inventory records in CAMO. The `id` is a unique identifier composed of three sub-elements that collectively represent the aircraft, assembly, or TRK inventory. If the `id` matches an existing asset in CAMO, the associated technical records are updated according to the request. If no match is found, a new asset is created along with its corresponding technical records.

Identifier information

Property name	Usage flags	UI mapping	Notes
asset	M	n/a	
asset .id	M, D	n/a	The asset.id may identify an aircraft (ACFT), a loose assembly (ASSY), or a loose tracked inventory (TRK). ARC processes all downstream elements (`currentConfiguration`, `inventor`, `completedTasksopenFaults`) relative to the specified root asset and its sub-components. If the referenced inventory is not ACFT, ASSY, or TRK, the upload fails validation and no records are processed.
asset.id .manufacturerCode	M, D	Inventory Details page. Inventory group box, Manufacturer	
asset.id .partNoOem	M, D	Inventory Details page. Inventory group box, OEM Part Number	
asset.id .serialNumber	M, U	Inventory Details page. Inventory group box, OEM Serial No	

Aircraft details

The `aircraftDetails` element enables the configuration of aircraft-specific properties. It is mandatory if inducting a new aircraft. If the element is not provided when updating an aircraft, the details associated with this element remain unchanged for the aircraft in CAMO.

If this element is provided, the following processing rules apply:

- When adding a new aircraft, the `aircraftDetails.registrationCode` property is mandatory and must be unique.
- When updating an existing aircraft, the `aircraftDetails` element is optional; but if provided, it must include the `registrationCode` property.
- If a value provided for a sub-element of the `aircraftDetails` element is not valid, an entry will be added to the validation log (see the *Upload result endpoint* details), and none of the values specified in the element are set on the aircraft.
- If this element is provided for a root asset that is not an aircraft, the entire element is ignored.

Aircraft-specific information

Property name	Usage flags	UI mapping	Notes
asset .aircraftDetails	O	Inventory Details page. Details tab, Aircraft Information section	Mandatory for creating an aircraft; optional for aircraft update. Ignored for non-aircraft assets.
asset.aircraftDetails .regulatoryBody	O, D	Inventory Details page. Details tab, Aircraft Information section, Regulatory Body	
asset.aircraftDetails .registrationCode	M, U	Inventory Details page. Details tab, Aircraft Information section, Registration Code	
asset.aircraftDetails .country	O, D	Inventory Details page. Details tab, Aircraft Information section, Country	
asset.aircraftDetails .capability	O, D	Inventory Details page. Details tab, Aircraft Information section, Capability field	

Property name	Usage flags	UI mapping	Notes
asset.aircraftDetails .variableNumber	O	Inventory Details page. Details tab, Aircraft Information section, OEM Variable No	
asset.aircraftDetails .lineNumber	O	Inventory Details page. Details tab, Aircraft Information section, OEM Line No	

Current configuration element

The `currentConfiguration` element is comprised of a list of position elements that describe the configuration of the root asset of the upload request by defining the tracked sub-components installed at specified positions on the asset. Understanding how to correctly populate these elements so that ARC can correctly install concrete inventory on an asset requires an understanding of how CAMO defines the configuration baseline of aircraft, sub-assemblies, and their tracked sub-components.

When the root asset is a TRK inventory, `currentConfiguration` describes the sub-structure beneath that root. Even when loose, TRK inventory must be associated with a configuration slot and corresponding part group of an assembly. Maintenix associates the TRK to an assembly configuration slot automatically unless explicitly controlled via the upload request.

Each `currentConfiguration` element may include an optional installation property, which allows the installation date of a sub-component to be set to its actual date of installation. This feature is particularly useful during induction scenarios, where the true installation date precedes the upload and accurate scheduling depends on reflecting that historical date. If this property is omitted and an installation is required to attach the sub-component, ARC assigns the current system time as the installation date.

If a sub-component already exists in CAMO and has an installation date, ARC and CAMO will not permit setting a new installation date that is earlier than the existing one. As a result, if a component is inducted without specifying an installation date, it may not be possible to backdate the installation later through another upload request. To avoid this issue, consider the following approaches:

1. Leave the position empty until the correct installation date is known, so ARC can record the date when it is known.

2. Provide installation dates in the upload request using a known historical reference point (such as the date of the sales agreement). More accurate dates can be updated in future uploads as needed.

Sub-component structure information

Property name	Usage flags	UI mapping	Notes
asset .currentConfiguration[]	O	Inventory Details page. Sub Inventory tab.	In Create mode, if not provided, CAMO will auto-populate mandatory positions with 'XXX' ("unknown") inventory. In Update mode, no updates will be made to the configuration.
asset.currentConfiguration[] .id	M	n/a	
asset.currentConfiguration[] .id .partGroupCode	M, D	Inventory Details page. Details tab, Inventory Information section, Part Group	
asset.currentConfiguration[] .id .positionName	O, D	Inventory Details page. Sub Inventory tab, Position column	If not provided, CAMO will select a position within the partGroup.
asset.currentConfiguration[] .inventoryId	O	n/a	If not provided, ARC will leave the position empty, removing inventory if necessary.
asset.currentConfiguration[] .inventoryId .manufacturerCode	M, D	Inventory Details page. Inventory group box, Manufacturer	
asset.currentConfiguration[] .inventoryId .partNoOem	M, D	Inventory Details page. Inventory group box, OEM Part Number	

Property name	Usage flags	UI mapping	Notes
asset.currentConfiguration[] .inventoryId .serialNumber	M, U	Inventory Details page. Inventory group box, OEM Serial No	
asset.currentConfiguration[] .nhAssembly	O	Inventory Details page. Inventory group box, Parent Assembly	Required if the position is on a sub-assembly of the root asset; otherwise, must be empty.
asset.currentConfiguration[] .nhAssembly .manufacturerCode	M, D	Inventory Details page. Inventory group box, Manufacturer	
asset.currentConfiguration[] .nhAssembly .partNoOem	M, D	Inventory Details page. Inventory group box, OEM Part Number	
asset.currentConfiguration[] .nhAssembly .serialNumber	M, U	Inventory Details page. Inventory group box, OEM Serial No	
asset.currentConfiguration[] .installation	O	n/a	
asset.currentConfiguration[] .installation .date	M	Inventory Details page. Historical tab, Installation tab, Event Date column	

TRK as Root Asset - A customer may use a TRK inventory as the root asset when inducting or synchronizing off-wing tracked components independently of an aircraft. Common scenarios include: *Engine modules tracked independently of aircraft installation, Rotables managed in pool or shop environments, Complex tracked inventory with its own sub-structure.* In these cases, the ARC upload message omits `aircraftDetails` and scopes all configuration, usage, tasks, and faults to the TRK inventory.

Inventory details element

The `inventoryDetails` array is a list of `Details` elements (see OpenAPI schema) that enable the capture of properties about the asset and its sub-components, particularly those properties that are used in maintenance scheduling.

Each `Details` element in the array is uniquely identified by its `inventoryId`.

If no value is provided for a property in the upload request (that is, the request does not include the property or the value is null, empty, or whitespace), no changes will be made, and the existing value will remain unchanged.

Asset and sub-component attribute information

Property name	Usage flags	UI mapping	Notes
asset .inventoryDetails[]	O	n/a	
asset.inventoryDetails[] .inventoryId	M, D	n/a	The asset/sub-component against which the details apply. The inventoryId must reference the asset or one of its sub-components that is already attached or will be attached by processing the currentConfiguration of the upload.
asset.inventoryDetails[] .inventoryId .manufacturerCode	M, U	Inventory Details page. Inventory group box, Manufacturer	
asset.inventoryDetails[] .inventoryId .partNoOem	M, D	Inventory Details page. Inventory group box, OEM Part Number	
asset.inventoryDetails[] .inventoryId .serialNumber	M, D	Inventory Details page. Inventory group box, OEM Serial No	

Property name	Usage flags	UI mapping	Notes
asset.inventoryDetails[] .applicabilityCode	O	Inventory Details page. Details tab, Inventory Information section, Applicability Code	Applies only to aircraft or ASSY-class assets. Ignored for TRK sub-components.
asset.inventoryDetails[] .releaseNumber	O	Inventory Details page. Details tab, Inventory Information section, Release Number	
asset.inventoryDetails[] .manufacturedDate	O, D	Inventory Details page. Details tab, Inventory Information section, Manufactured Date	Required if the inventory has a scheduling rule based on the manufactured date. If both manufactured and received date properties are provided, the receivedDate must be at least one second later than the manufacturedDate.
asset.inventoryDetails[] .receivedDate	O, D	Inventory Details page. Details tab, Inventory Information section, Received Date	Required if the inventory has a scheduling rule based on the received date. If both manufactured and received date properties are provided, the receivedDate must be at least one second later than the manufacturedDate.
asset.inventoryDetails[] .operator	O, D	Inventory Details page. Details tab, Inventory Information section, Operator	Applies only to aircraft assets. Ignored for ASSY or TRK sub-components. Must map to a valid operator in CAMO.

Property name	Usage flags	UI mapping	Notes
asset.inventoryDetails[] .originalEquipment	O		Must map to a valid component in CAMO, either pre-existing or included in the upload current configuration.
asset.inventoryDetails[] .originalEquipment .manufacturerCode	M, D	Inventory Details page. Inventory group box, Manufacturer	
asset.inventoryDetails[] .originalEquipment .partNoOem	M, D	Inventory Details page. Inventory group box, OEM Part Number	
asset.inventoryDetails[] .originalEquipment .serialNumber	M, D	Inventory Details page. Inventory group box, OEM Serial No	

Inventory usages element

The `inventoryUsages` array is a list of `inventoryUsage` elements (see OpenAPI schema definition) that capture current usage information for the asset and its sub-components.

Each `InventoryUsage` in the array is uniquely identified by its `inventoryId`. The `InventoryUsage` then includes a list of current usages to be captured against that inventory.

Each usage parameter must already be defined in CAMO against the inventory (by part or by position). At a minimum, the usage values must include a TSN.

Asset and sub-component current usage information

Property name	Usage flags	UI mapping	Notes
asset .inventoryUsages[]	O	Inventory Details page. Details tab, Current Usage section	If the list is not provided or is empty, no usage corrections will be applied to the asset or its sub-components.

Property name	Usage flags	UI mapping	Notes
asset.inventoryUsages[] .inventoryId	M, D	n/a	The asset/sub-component against which the current usage is being captured. The inventoryId must reference the asset or one of its sub-components that is already attached or will be attached by processing the currentConfiguration of the upload.
asset.inventoryUsages[] .inventoryId .manufacturerCode	M, D	Inventory Details page. Inventory group box, Manufacturer	
asset.inventoryUsages[] .inventoryId .partNoOem	M, D	Inventory Details page. Inventory group box, OEM Part Number	
asset.inventoryUsages[] .inventoryId .serialNumber	M, U	Inventory Details page. Inventory group box, OEM Serial No	
asset.inventoryUsages[] .usages[]	O		
asset.inventoryUsages[] .usages[] .parameterCode	M	Inventory Details page. Details tab, Current Usage section, Usage Parm	The specified parameter must already exist for the specified inventory. If it does not, the usage will not be applied.
asset.inventoryUsages[] .usages[] .tsn	M	Inventory Details page. Details tab, Current Usage section, TSN	

Property name	Usage flags	UI mapping	Notes
asset.inventoryUsages[] .usages[] .tsi	O	Inventory Details page. Details tab, Current Usage section, TSI	
asset.inventoryUsages[] .usages[] .tso	O	Inventory Details page. Details tab, Current Usage section, TSO	

Completed tasks element

The `completedTasks` array is a list of task elements that capture information about maintenance tasks that were completed in the external system and create historic tasks in CAMO. The tasks provided in this list are expected to correspond to baseline BLOCK or REQ task definitions defined against the baseline assembly of the root asset or its sub-components.

Each `completedTask` in the array is uniquely identified by the combination of:

- `inventoryId` identifies the inventory item against which the task was completed (either the root asset or one of its sub-components).
- `code` is the CAMO task code of the completed task.

As a completed task, this element must also provide (at a minimum) the `completionDate` of the task. The table below identifies some additional optional information that may be provided as part of each `completedTask`.

Details about how ARC matches completed tasks to task definitions and to existing tasks, and applies usages-at-completion are defined in the Asset Record Coordination section of the *User Guide*.

Historic task information

Property name	Usage flags	UI mapping	Notes
asset .completedTasks[]	O		If the list is not provided, no additional historic tasks will be created against this asset.

Property name	Usage flags	UI mapping	Notes
asset.completedTasks[] .inventoryId	M, D	n/a	The identifier of the asset/sub-component against which the task will be completed. The inventoryId must reference the asset or one of its sub-components that is already attached or will be attached by processing the currentConfiguration of the upload.
asset.completedTasks[] .inventoryId .manufacturerCode	M, D	Inventory Details page. Inventory group box, Manufacturer	
asset.completedTasks[] .inventoryId .partNoOem	M, D	Inventory Details page. Inventory group box, OEM Part Number	
asset.completedTasks[] .inventoryId .serialNumber	M, U	Inventory Details page. Inventory group box, OEM Serial No	
asset.completedTasks[] .code	M	Task Details page. Task title	
asset.completedTasks[] .completionDate	M	Task Details page. Task Information tab, Task Information section, Actual End Date	
asset.completedTasks[] .documentReference	O	Task Details page. Task Information tab, Technical References section	

Property name	Usage flags	UI mapping	Notes
asset.completedTasks[] .externalKey	O	n/a	An optional key mapping the CAMO task to a task in an external system
asset.completedTasks[] .action	O	Task Details page. Task Execution tab, Action section	
asset.completedTasks[] .usagesAtCompletion[]	O	Task Details page. History tab, Usage At Completion section	
asset.completedTasks[] .usagesAtCompletion[] .parameter	M, D		Elements of the usagesAtCompletion array must include a parameter code that maps to a usage parameter being tracked against that inventory (either by part or position)
asset.completedTasks[] .usagesAtCompletion[] .tsn	M		Elements of the array must include a value for TSN.
asset.completedTasks[] .usagesAtCompletion[] .tsi	O		If not provided, this value will be set to the value of the TSN.
asset.completedTasks[] .usagesAtCompletion[] .tso	O		If not provided, this value will be set to the value of the TSN.

Open faults element

The `openFaults` array is a list of fault elements that reflect the current, complete list of open and deferred faults recorded against the root assembly and its sub-components. Faults with an empty or blank logbook reference are not processed. For each remaining fault in the array, ARC will search for a matching fault in CAMO and update the existing fault if found, otherwise raise a

new fault. If the root asset has open/deferred faults in CAMO that are not in the list, ARC will close them as No Fault Found.

If the `openFaults` array is not provided or is a 0-length array, fault processing will be bypassed and no modifications will be made to existing faults in CAMO.

Each `Fault` in the array is compared to existing faults in CAMO using the combination of:

- `inventoryId` of the asset, sub-assembly, or sub-component against which the fault was raised. System faults use the asset/sub-assembly as the inventory identifier.
- `logbookReference` is the logbook reference captured in CAMO.
- `failedSystem` is used for faults which are raised against an assembly system (configuration slot) rather than directly against the assembly/sub-component inventory.

Each element in the array must also provide (at a minimum) the following properties of the fault which are required by CAMO for tracking and scheduling purposes:

- name
- foundOnDate
- severity

Details about how ARC matches current faults to known faults are defined in the Asset Record Coordination section of the *User Guide*. The table below identifies some additional optional information that may be provided as part of each fault.

Asset and sub-component open fault information

Property name	Usage flags	UI mapping	Notes
asset .openFaults[]	O		If the list is empty or not provided, fault processing will be bypassed. No updates will be applied to any existing faults in CAMO.
asset.openFaults[] .inventoryId	M, D	n/a	The inventory asset or sub-component against which the fault is raised. The inventoryId must reference the asset or one of its sub-components that is already attached or will be attached by processing the currentConfiguration of the upload.

Property name	Usage flags	UI mapping	Notes
asset.openFaults[] .inventoryId .manufacturerCode	M, D	Inventory Details page. Inventory group box, Manufacturer	
asset.openFaults[] .inventoryId .partNoOem	M, D	Inventory Details page. Inventory group box, OEM Part Number	
asset.openFaults[] .inventoryId .serialNumber	M, U	Inventory Details page. Inventory group box, OEM Serial No	
asset.openFaults[] .logbookReference	O	Fault Details page. Details group box, Logbook Reference	Used for matching the fault against existing CAMO faults. If not provided, a new fault will be raised.
asset.openFaults[] .failedSystem	O, D	Fault Details page. Task Information tab, Fault Details section, Config Slot	Used for matching the fault against existing CAMO faults. If not specified when creating a new fault, the fault will be created against the specified inventoryId rather than a system (configuration slot).
asset.openFaults[] .documentReference	O	Fault Details page. Task Information tab, Technical References section, Document Reference	
asset.openFaults[] .foundOnDate	M	Fault Details page. Fault Information tab, Fault Details section, Found On Date	

Property name	Usage flags	UI mapping	Notes
asset.openFaults[] .dueDate	O, D	Fault Details page. Details section, Due	This property is required if the fault is deferred using a MEL deferral, and ignored otherwise.
asset.openFaults[] .deferralClass	O, D	Fault Details page. Fault Information tab, Fault Details section, Deferral Class	This value must be defined in REF_FAIL_DEFER. This property is required if the fault is deferred using a MEL deferral, and ignored otherwise.
asset.openFaults[] .deferralReference	O	Fault Details page. Fault Information tab, Fault Details section, Deferral Reference	This property is required if the fault severity is MEL. This property is required if the fault is deferred using a MEL deferral, and ignored otherwise.
asset.openFaults[] .name	M	Fault Details page. Task title.	
asset.openFaults[] .description	O	Fault Details page. Task Execution tab, Description section	
asset.openFaults[] .source	O, D	Fault Details page. Fault Information tab, Fault Details section, Fault Source	This value must be defined in REF_FAULT_SOURCE .fault_source_cd. If not provided when creating a new fault, this value will default to "MESSAGE."

Property name	Usage flags	UI mapping	Notes
asset.openFaults[] .severity	M, D	Fault Details page. Fault Information tab, Fault Details section, Fault Severity	This value must be defined in REF_FAIL_SEV.fail_sev_cd. If the severity is MINOR or MEL, the fault will be automatically deferred. If an invalid value is provided when creating a new fault, this value will default to "UNKNOWN."
asset.openFaults[] .priority	O, D	Fault Details page. Task Information tab, Task Information section, Priority	This value is defined in REF_TASK_PRIORITY .task_priority_cd.
asset.openFaults[] .externalKey	O	n/a	This optional value can be used to map a CAMO fault to its equivalent fault in an external system.
asset.openFaults[] .dueValues[]	O	Fault Details page. Scheduling tab, Due section	If the fault deadline is based on usages, this array defines those usage-based deadlines. This array is ignored if the severity is not MEL.
asset.openFaults[] .dueValues[] .parameterCode	M, D	Fault Details page. Scheduling tab, Due section, Parameter column	
asset.openFaults[] .dueValues[] .value	M	Fault Details page. Scheduling tab, Due section, Interval column	
asset.openFaults[] .usagesWhenFound[]	O	Fault Details page. History tab, Usages When Found section	

Property name	Usage flags	UI mapping	Notes
asset.openFaults[] .usagesWhenFound[] .parameter	M, D		
asset.openFaults[] .usagesWhenFound[] .tsn	M		
asset.openFaults[] .extension	O, D	Fault Details page. Scheduling tab, Due section	Use this element to apply a deadline extension to the fault. Note that an extension can only be applied if the fault severity is MEL.
asset.openFaults[].extension .dueDate	O	Fault Details page. Scheduling tab, Due section, Est. Date column	
asset.openFaults[].extension .note	O	Fault Details page. History tab, History section, Note column	
asset.openFaults[].extension .referenceNumber	O	Fault Details page. History tab, History section, Note column	
asset.openFaults[].extension .dueValues[]	O	Fault Details page. Scheduling tab, Due section	This array is ignored if the severity is not MEL.
asset.openFaults[].extension .dueValues[] .parameterCode	M, D	Fault Details page. Scheduling tab, Due section, Parameter column	Note that only usage parameters associated with an existing dueValue can be extended.
asset.openFaults[].extension .dueValues[] .value	M	Fault Details page. Scheduling tab, Due section, Interval column	

Response details

The Upload API responds asynchronously with a status representation that reflects the current state of the request. Further details of the status representation can be found in *Upload status*.

Error responses

When an upload request cannot be processed, ARC responds with a problem details representation that includes the HTTP status code as well as a more granular processing error code. The full specification of the response format is detailed in *ARC error responses*.

The table below lists the HTTP status code and ARC-specific error code pairs that may occur in response to an upload request error. See also *Standard error codes* for additional errors that may be returned by all endpoints.

Error responses

HTTP status code	ARC error code	Description
400 Bad Request	API-0002	This error occurs if the request payload was not provided (content-length is 0).
400 Bad Request	API-0004	This error occurs when the request payload cannot be de-serialized due to syntactic issues. Common causes include malformed JSON, incorrect data types, or structural mismatches with the expected schema.

Upload status

The upload status resource provides a snapshot of the current state of an ARC upload request. This endpoint should be used to poll the status of an upload until it completes and its associated validation log becomes available.

Sample request/response

Request

`GET https://<host:port>/arc/api/uploads/status/{uploadId}`

Success response

200 Ok:

```
{
    "uploadId": "19070017ADCF11F0B1E24851C5518011",
    "workflow": "VALIDATE_AND_PROCESS",
    "status": "PROCESSED",
    "receivedDate": "2025-10-20T16:09:03Z",
    "completedDate": "2025-10-20T16:09:06Z",
    "externalReference": "ext-ref-1234"
}
```

Response details

The status representation includes metadata about the upload request that enables consumer applications to track progress and correlate activity across systems.

Property name	Usage flags	UI mapping	Notes
uploadId	M	n/a	System-generated unique identifier for the upload request.
workflow	M	n/a	
status	M	ARC Message Management page. Message View tab	
receivedDate	M	n/a	
completedDate	O	n/a	Will be null until the upload request has completed processing. While the upload is incomplete, it cannot be queried through the upload result API, nor will it be visible on the ARC Management page.
externalReference	O	ARC Message Management page. Message View tab, External Reference column	If provided in the upload request, this value will be in the response.

Error responses

When a request to an ARC API endpoint cannot be processed, ARC responds with a problem details representation that includes the HTTP status code as well as a more granular processing error code. The full specification of this format is detailed in *ARC error responses*.

The following table identifies the HTTP status code and ARC-specific error code pairs that may occur in response to sending a request to the upload status endpoint. See also *Standard error codes* for additional errors that may be returned by all endpoints.

Error responses

HTTP status code	ARC error code	Description
404 Not Found	API-0005	The URL path could not be resolved. This occurs when the path is incorrect or is missing the uploadId.
404 Not Found	ARC-0002	The URL path is correct, but a matching upload request could not be found.

Upload result

This endpoint is used to query the processing details/outcomes of the upload request.

Sample request/response

Request

`GET https://<host:port>/arc/api/uploads/result/{uploadId}`

Success response

`200 Ok`:

```
{
    "uploadStatus": {
        "uploadId": "DC2E5EC7ADCE11F0B1E24851C5518011",
        "workflow": "VALIDATE_AND_PROCESS",
        "status": "PROCESSEDWITHWARNING",
        "receivedDate": "2025-10-20T16:07:21Z",
        "completedDate": "2025-10-20T16:07:23Z",
        "externalReference": "ext-ref-1234"
    },
    "validationLog": [
        {
            "entryType": "asset",
```

```
            "id": "DC6DFFE7ADCE11F0B1E24851C5518011",
            "description": "Aircraft Asset for A319/A320/ABI:33410 was
 updated successfully.",
            "severity": "INFO",
            "inventoryId": {
                "serialNumber": "33410",
                "manufacturerCode": "ABI",
                "partNoOem": "A319/A320"
            }
        },
        {
            "entryType": "inventoryDetail",
            "id": "DCC74378ADCE11F0B1E24851C5518011",
            "description": "Inventory Details for CFM56-3C/
CFM:SN000279 was updated successfully.",
            "severity": "INFO"
        },
        {
            "entryType": "inventoryUsage",
            "id": "DCCA77C9ADCE11F0B1E24851C5518011",
            "description": "The usage parameter code AOC is not
 applicable on inventory CFM56-3C/CFM:SN000279. As a result, this
 usage parameter was ignored for the inventory. Verify the inventory
 usage in Maintenix, and correct if necessary.",
            "severity": "WARNING"
        },
        {
            "entryType": "asset",
            "id": "DD16C30EADCE11F0B1E24851C5518011",
            "description": "The asset was processed successfully.",
            "severity": "SUCCESS",
            "inventoryId": {
                "serialNumber": "33410",
                "manufacturerCode": "ABI",
                "partNoOem": "A319/A320"
            }
        },
        {
            "entryType": "completedTask",
            "id": "DCD9BA0BADCE11F0B1E24851C5518011",
            "description": "A completed ad hoc task Eng Task 1 for
 inventory CFM56-3C/CFM:SN000279 already exists. As a result, an
 additional completed task was not created.",
            "severity": "INFO"
        },
        {
            "entryType": "openFault",
            "id": "DD164DDDADCE11F0B1E24851C5518011",
            "description": "Open fault Eng Fault 1 for inventory
 CFM56-3C/CFM:SN000279 was created successfully.",
            "severity": "INFO"
        },
        {
            "entryType": "inventoryUsage",
            "id": "DCCB3B1AADCE11F0B1E24851C5518011",
```

```
            "description": "The usage parameter code AOH is not
 applicable on inventory CFM56-3C/CFM:SN000279. As a result, this
 usage parameter was ignored for the inventory. Verify the inventory
 usage in Maintenix, and correct if necessary.",
            "severity": "WARNING"
        }
    ]
}
```


Response details

The upload result representation conveys the outcome of the ARC upload request in full, including the overall status and a detailed list of findings generated by ARC and CAMO during processing. This list, known as the Validation Log, captures processing failures, validation failures, warnings where operations could not be performed, informational advisories where ambiguities in the request data were interpreted (such as defaulting null values), and confirmations of some operations that were processed successfully. Each of these types of log entries are associated with a severity to indicate their nature.

Entries in the validation log are polymorphic: they have the same base representation and use a discriminator property to identify the specific type of entry. Currently, only `asset`-type entries have a type-specific representation, which includes the `inventoryId` of the root asset specified in the original request.

Property name	Usage flags	UI mapping	Notes
uploadId	M	n/a	System-generated unique identifier for the upload request.
workflow	M	n/a	
status	M	ARC Message Management page. Message View tab	
receivedDate	M	n/a	
completedDate	O	n/a	
externalReference	O	ARC Message Management page. Message View tab, External Reference column	If provided in the upload request, this value will be in the response.

Property name	Usage flags	UI mapping	Notes
validationLog	M	ARC Process Log page	
validationLog .entityType	M	ARC Process Log page, Type column	One of: - message - asset - currentConfiguration - inventoryDetail - inventoryUsage - completedTask - openFault
validationLog .id	M	n/a	Internally generated system identifier of the log entry.
validationLog .description	M	ARC Process Log page, Description column	A localizable user-friendly string describing the finding.
validationLog .severity	M	ARC Process Log page, Severity column	One of: - SUCCESS - INFO - WARNING - ERROR

Asset log entry details

Asset log entries extend the base log entry to include the `inventoryId` of the root asset.

Property name	Usage flags	UI mapping	Notes
validationLog .inventoryId	M	Inventory Details	

Property name	Usage flags	UI mapping	Notes
validationLog.inventoryId .manufacturerCode	M	n/a	
validationLog.inventoryId .partNoOem	M	ARC Message Management page. Message View tab	
validationLog.inventoryId .serialNumber	M	n/a	

Error responses

When a request to an ARC API endpoint cannot be processed, ARC responds with a problem details representation that includes the HTTP status code as well as a more granular processing error code. The full specification of this format is detailed in *ARC error responses*.

The following table identifies the HTTP status code and ARC-specific error code pairs that may occur in response to sending a request to the upload result endpoint. See also *Standard error codes* for additional errors that may be returned by all endpoints.

Error responses

HTTP status code	ARC error code	Description
404 Not Found	API-0005	The URL path could not be resolved. This occurs when the path is incorrect or is missing the uploadId.
404 Not Found	ARC-0002	The URL path is correct but a matching upload request could not be found.
404 Not Found	ARC-0003	Upload is still in progress. The URL path is valid, and ARC-0003 indicates that the specified upload has been identified, but the upload result is not available for retrieval.

ARC health check

This endpoint is used to query the health of the ARC API.

Sample request/response

Request

`GET https://<host:port>/arc/api/health`

Success response

`200 Ok`:

```
{
    "currentTime": "2025-10-20T16:26:34Z",
    "message": "ARC API is up and running."
}
```

Response details

Property name	Usage flags	UI mapping	Notes
currentTime	M	n/a	
message	M	n/a	

ARC error responses

ARC API error responses use a consistent, standardized structure to communicate issues encountered during request processing. When a request fails – whether due to validation errors, unsupported operations, or internal exceptions – the API returns a response with media type `application/problem+json` (originally defined in RFC 9457). This structured response ensures consistency across ARC APIs and provides clear diagnostic information to help consumer applications identify, categorize and respond to errors effectively.

Sample error response

The error response contains the following properties:

```
{
   "status": 404,
   "detail": "The specified upload request is still in progress.
 Reference 0123456789ABCDEFABCDEF0123456789",
   "message": "11112222333344445555666677778888 is still in progress",
   "errorCode": "ARC-0003",
   "correlationId": "0123456789ABCDEFABCDEF0123456789"
}
```

Response details

ARC error response format

Property name	Data type	RFC 9457 vs Custom	Description
status	int	RFC 9457	The HTTP status code generated by the server for this occurrence of the problem.
detail	string	RFC 9457	A human-readable explanation specific to this occurrence of the problem. The "detail" string, if present, attempts to focus on helping the client correct the problem. In the ARC API, these explanations describe problems in the interactions between two systems, not application-level business logic errors. They indicate a problem in the way that the HTTP request was formulated by the consumer or handled by the server. They are not intended for end-user presentation beyond including enough information so that the user can provide some context to the support organization (such as a correlationId, to help with application log retrieval). As a consequence, they are not localizable.
message	string	Custom	A more developer-oriented message that relays technical information about the problem, that may help with debugging.
errorCode	string	Custom	A more specific identifier of the type of problem encountered. For example, this value can help to distinguish an "Invalid path" from an "Invalid path parameter" (i.e. invalid uploadId) error. ARC Error Codes are listed above.

Property name	Data type	RFC 9457 vs Custom	Description
correlationId*	string	Custom	Identifier assigned to the HTTP request. If the consumer application provided a correlationId in the request, the value will be provided here to facilitate debugging. If no correlationId is provided, then one will be generated by CAMO. Log files within CAMO generally include the correlationId. *Note: The correlationId is not included when the error is caused by an Authorization failure (i.e 403 Forbidden).

Standard error codes

HTTP status code	ARC error code	Description
403 Forbidden	API-0001	The requesting user presented in the Authorization header is valid but is not authorized to access the ARC API. The two permissions controlling access to ARC API resources are: • API_ARC (operational endpoints) • API_VIEW_REQUEST (OpenAPI specification)
405 Method Not Allowed	API-0007	The URL of the request is valid, but does not support the verb used in the request (e.g. used a POST instead of GET).
406 Not Acceptable	API-0006	The ARC API provides responses using `application/json`. This error indicates that the Accept header provided in the request specifies a media type not supported by the endpoint.

HTTP status code	ARC error code	Description
415 Unsupported Media Type	API-0008	The ARC API accepts requests with Content-Type `application/json`. This error occurs when the consumer's request header specifies a different media type.

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.