# IFS Aviation Maintenance

## Actuals Loader User Guide

ICAM 26.2



**Document Revision: 26.2-1**

**Publication Date: November 03, 2026**

# Contents


1 Purpose ..... 6

2 Overview of Actuals Loader ..... 7

Actuals Data ..... 7

3 Loading inventory ..... 10

Loading unknown sub-components ..... 11

Loading inventory .csv files ..... 11

Populating inventory staging tables ..... 13

C_RI_INVENTORY (Root inventory) ..... 13

C_RI_INVENTORY - ACFT ..... 14

C_RI_INVENTORY - ASSY ..... 16

C_RI_INVENTORY - TRK ..... 17

C_RI_INVENTORY - SER ..... 18

C_RI_INVENTORY - BATCH ..... 19

C_RI_INVENTORY - KIT ..... 20

C_RI_INVENTORY_SUB (Sub Inventory) ..... 20

C_RI_INVENTORY_SUB - TRK ..... 21

C_RI_INVENTORY_SUB - SER ..... 22

C_RI_INVENTORY_USAGE (Inventory Usage) ..... 24

C_RI_INVENTORY_CAP_LEVELS ..... 25

C_RI_ATTACH (Inventory attachments) ..... 25

Inventory data processing tables ..... 26

Validating inventory data ..... 26

Validating inventory data using .bat file ..... 27

Validating inventory data using packages ..... 28

Error handling for inventory data ..... 28

Importing inventory data ..... 28

Importing inventory data using .bat file ..... 29

Importing inventory data using packages ..... 29

Finalizing inventory data ..... 30

Finalizing inventory data using .bat file ..... 30

Finalizing inventory data using procedure ..... 30

4 Loading historical usage....31
Loading historical usage using .csv files....31
Populating historical usage staging tables....32
AL_HISTORICAL_USAGE....32
Historical usage data processing tables....33
Validating historical usage data....33
Validating historical usage data using .bat file....34
Validating historical usage data using packages....34
Error handling for usage data....35
Importing historical usage data....35
Importing historical usage data using .bat file....35
Importing historical usage data using packages....35
5 Loading tasks....37
Loading task .csv files....38
Populating task staging tables....39
C_RI_TASK (task)....39
C_RI_TASK_SHED....41
Task data processing tables....43
Validating task data....44
Validating task data using .bat file....44
Validating task data using packages....45
The Missing Task Report....45
Error handling for task data....47
Importing task data....47
Importing task data using .bat file....47
Importing task data using packages....47
6 Loading faults....49
Populating fault staging tables....49
AL_OPEN_DEFERRED_FAULT....49
Fault data processing tables....52
Validating fault data....53
Validating fault data using .bat file....53
Validating fault data using packages....54

Error handling for fault data....54
Importing fault data....55
Importing fault data using .bat file....55
Importing fault data using packages....55

7 Loading work packages....56
Populate Work Package Staging Tables....56
AL_WORK_PACKAGE....56
AL_WORK_PACKAGE_TASK....58
AL_WORK_PACKAGE_TASK_ATA....59
Work package data processing tables....60
Validating work package data....60
Validating work package data using .bat file....60
Validating work package data using packages....60
Error handling for work packages....61
Importing work package data....61
Importing work package data using .bat file....61
Importing work package data using packages....61

8 Loading purchase orders....63
Loading purchase orders .csv files....63
Populating purchase order staging tables....64
C_PO_HEADER....64
C_PO_LINE....66
C_PO_LINE_TAX....68
C_PO_LINE_CHARGE....68
Purchase order data processing tables....68
Validating purchase order data....69
Validating purchase order data using .bat file....69
Validating purchase order data using packages....70
Error handling for purchase order data....70
Importing purchase order data....70
Importing purchase order data using .bat file....70
Importing purchase order data using packages....71

9 Loading inventory original equipment....72

Loading inventory original equipment .csv files....72
Populating inventory original equipment staging table....73
AL_IOE_ASSMBL....73
Inventory original equipment data processing tables....74
Validating inventory original equipment data....75
Validating inventory original equipment using .bat file....75
Validating inventory original equipment data using packages....76
Error handling for inventory original equipment data....76
Importing inventory original equipment data....76
Importing inventory original equipment data using .bat file....76
Importing inventory original equipment data using packages....77

# 1 Purpose

This guide is intended for Data Migration specialists or members who are responsible for, or involved in, the migration of data from legacy systems within your organization to the CAMO database.

# 2

# Overview of Actuals Loader

Loading actuals data consists of migrating technical maintenance data out of legacy systems, from paper, or electronic documents, and then loading that data into CAMO. Once actuals data is loaded, personnel in various departments can use CAMO to plan and track the maintenance of your aviation assets.

Actuals data refers to the technical maintenance data for aircraft and serialized components: such as the serial number, manufactured date, and accumulated usage; the list of sub-components; the maintenance tasks that have been completed and the tasks that are upcoming; the deferred faults; and so on.

## Actuals Data

The Actuals Loader tool validates the aircraft's current technical records against the configuration and maintenance program rules that are loaded into CAMO as part of the baseline data. You must load the baseline data into CAMO prior to loading actuals data. The actuals data contains the following information:

- Aircraft details, such as the serial number, accumulated usage, and current location of the aircraft
- Off-wing component details, such as the serial number, accumulated usage, and current location of the component
- Sub-component details, such as the serial number and part number of every installed component and their position
- Maintenance history, such as the date and usage-at-completion of every completed requirement on the aircraft and its sub-components
- Open deferred faults
- Open work packages
- Open and issued purchase orders

- The original equipment details for inventory

The Actuals Loader can be used during the initial CAMO implementation to load all of the aircraft and off-wing inventory into the system. This happens before CAMO has been turned on, and before users are actively logged into the system. The Actuals Loader can also be used after the system is up and running, to load newly received aircraft or off wing components. This allows you to load new aircraft data, even while users are actively logged in and interacting with CAMO. To improve database performance, Gather Stats should be ran prior to executing Actuals Loader tools.

At a high level, the process for loading actuals data is as follows:

1. Install Data Loader
   a. Verify that the software requirements and prerequisite conditions are met.
   b. Install the Actuals Loader staging processing tables in your CAMO database.
   c. Prior to Actual loader tools, execute Gather Stats against the database schema
2. Load inventory
   a. Populate the staging tables with the inventory data.
   b. Validate the inventory data in the staging tables. This is an iterative process; you validate the data in the staging tables, resolve the errors in the data, and validate again. You repeat the process until the validation succeeds.
   c. Import the successfully validated inventory data into CAMO.
3. Load inventory historical usage
   a. Populate historical usage staging tables
   b. Validate historical usage data
   c. Import historical usage data
4. Load tasks
   a. Populate task staging tables
   b. Validate task data
   c. Import task data. (You can load tasks in multiple stages, where each time you are trying to load a different group of tasks. For example, you could load your checks in the first pass, your component requirements in the 2nd pass, and your MOD tasks in the 3rd pass.)
5. Load faults
   a. Populate fault staging tables
   b. Validate fault data
   c. Import fault data
6. Load work packages
   a. Populate work package staging tables
   b. Validate work package data
   c. Import work package data

7. Load open and issued purchase orders
   a. Populate purchase orders staging tables
   b. Validate purchase orders data
   c. Import purchase orders data
8. Load inventory original equipment
   a. Populate original equipment staging tables
   b. Validate original equipment data
   c. Import original equipment data
9. Finalize inventory
   a. As a final step, there are certain data cleanup activities that must happen on the loaded inventory. This step must happen after all of the tasks, faults and work packages have been loaded.

**Note** If an aircraft or root inventory in the staging table is missing sub-components, Actuals Loader auto-creates the missing items when the complete.assy.bool parameter is set to FULL. To load incomplete inventory, or use the CAMO Ready for Build feature, set the complete.assy.bool parameter to NOT.

# 3

# Loading inventory

After your baseline data is loaded in the CAMO database, you can load the data for your inventory items. The following lists the classifications of inventory used by CAMO:

- ACFT – Aircraft assembly
- ASSY – Non-aircraft assembly
- TRK – Tracked inventory items; items that are positionally tracked when installed
- SER –Serialized items; items that are serialized, but not positionally tracked when installed
- BATCH – Batch controlled items
- KIT - A collection of items used together to carry out a specific task

Although common staging tables are used to stage data for all classifications of inventories, there are differences in how the columns should be filled out for each classification.

The following sections define the customized use of this Oracle table for each inventory classification, particularly in terms of discussion around which attributes are mandatory, which are optional, and finally, those that simply are not applicable.

**Note** Since the ACFT, ASSY, TRK, SER, and BATCH types of inventory are all staged in common staging tables (C_RI_INVENTORY and C_RI_INVENTORY_SUB), the tables contain a "super set" of the possible attributes for each of these five (5) types of inventory.

The sections below only describe those columns that are relevant to the given inventory class. If a column exists on the staging table but the column is not listed in the section below, then the column is not applicable to that class of inventory and should not be populated with data. For example, the column PRIVATE_BOOL on C_RI_INVENTORY is only used when loading inventory of class ACFT, and should not be populated for other classes of inventory.

## Loading unknown sub-components

**About this task**

Sometimes the legacy system that is the source of your data does not track the Serial Numbers of all installed subcomponents. In CAMO, these unknown sub-components will be loaded using a Serial Number of "XXX".

However, although the legacy system does not know the Serial Number of the sub-component, it does know the maintenance information for those installed components (last done and/or next due). This is typically because the legacy system tracked those maintenance items against the aircraft, but in CAMO those tasks need to be tracked against the sub-component inventory.

In order to load maintenance against those unknown sub-components, you must do the following:

1. Load the sub-components into the C_RI_INVENTORY_SUB table, but invent a unique SERIAL_NO_OEM for the sub-component using the prefix "AL_XXX_". A useful practice for this unique serial number is "AL_XXX_<unique id>" where <unique id> would be the aircraft tail number + the component type. For example: "AL_XXX_741_FLIGHT_BEACON".
2. Load the tasks in the C_RI_TASK table (as described in the Tasks section below) and provide this invented serial number in the task's SERIAL_NO_OEM column.
3. Once the tasks have all been loaded, perform the Finalize Inventory step (described in the section below), which will convert those AL_XXX sub-components into "XXX" in the CAMO database.

## Loading inventory .csv files

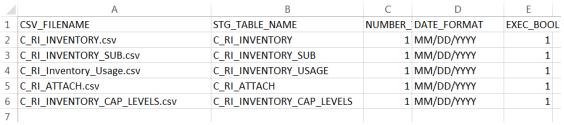
In order to use the Actuals Loader, you must populate the staging tables described in the next section. Instead of using SQL to fill in these staging tables, Actuals Loader provides the option to fill in .csv files and then load those .csv files into the staging tables.

**Steps**

1. Fill out your C_RI_*.csv files following these example templates found in `<installation_folder>\dataload\script_Actuals\inventory\data`
   - C_RI_INVENTORY_sample.csv
   - C_RI_INVENTORY_SUB_sample.csv
   - C_RI_INVENTORY_USAGE_sample.csv
   - C_RI_ATTACH_sample.csv
   - C_RI_INVENTORY_CAP_LEVELS_sample.csv
2. Copy your completed files to the same directory (`<installation_folder>\dataload>\script_Actuals\inventory\data`)
3. The directory will now contain your files:

- C_RI_INVENTORY.csv
- C_RI_INVENTORY_SUB.csv
- C_RI_INVENTORY_USAGE.csv
- C_RI_ATTACH.csv
- C_RI_INVENTORY_CAP_LEVELS.csv

4. Update columns A, C, D and E of the file `<installation_folder>\dataload>\script_Actuals\inventory\data\inventory_load_config.csv` with the details of your C_RI_*.csv files.
5. This file contains configuration information for the inventory loading script. The figure below shows an example of a completed `inventory_load_config.csv` file.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | CSV_FILENAME | STG_TABLE_NAME | NUMBER_ | DATE_FORMAT | EXEC_BOOL |
| 2 | C_RI_INVENTORY.csv | C_RI_INVENTORY | 1 | MM/DD/YYYY | 1 |
| 3 | C_RI_INVENTORY_SUB.csv | C_RI_INVENTORY_SUB | 1 | MM/DD/YYYY | 1 |
| 4 | C_RI_Inventory_Usage.csv | C_RI_INVENTORY_USAGE | 1 | MM/DD/YYYY | 1 |
| 5 | C_RI_ATTACH.csv | C_RI_ATTACH | 1 | MM/DD/YYYY | 1 |
| 6 | C_RI_INVENTORY_CAP_LEVELS.csv | C_RI_INVENTORY_CAP_LEVELS | 1 | MM/DD/YYYY | 1 |
| 7 | | | | | |

   - **CSV_FILENAME** (column A): List each C_RI_*.csv file that you would like to load into the staging tables. Make sure that none of these are blank, otherwise the loading script will fail.
   - **STG_TABLE_NAME** (column B): The corresponding Oracle staging table name. Do not change these values.
   - **NUMBER_TITLE_ROWS** (column C): List the number of rows used for titles in your file from column A. If the data in your C_RI_*.csv data file does not start at the 2nd row, increment this number to show how many title rows to skip before the inventory loading script looks for your data.
   - **DATE_FORMAT** (column D): List the format of the dates (MANUFACT_DT, RECEIVED_DT, etc.) used in your file from column A. All dates must have consistent format within each C_RI_*.csv file, otherwise the validation will have errors.
   - **EXEC_BOOL** (column E): This is a binary value. If the value is a 1, the data loading script will load the file's data into the staging tables. If the value is a 0, the data loading script will skip over this file and will not load it. (In the example in the figure above, the C_RI_ATTACH.csv file will not be loaded.)
6. Load the inventory files into the staging tables by running the script file `<installation_folder>\dataload\script_Actuals\inventory\a.load_inventory.bat`.
7. When the script completes, it displays a browser window showing an Actuals Loader report containing the status of the loading process. The report has a table containing the following fields:
   - **Staging Table**: Lists each C_RI_* inventory table name.
   - **Record Count**: Indicates how many records were loaded into each table. In the figure above, the last Record Count value is 0 because the file was skipped as per the EXEC_BOOL value of 0 in your inventory_load_config.csv file as per step 3) above.

Listed below are some common issues encountered during loading inventory and their possible causes:

| Issue | Possible Cause |
|---|---|
| The Record Count in the report is a 0 but you didn't set EXEC_BOOL to 0. | In inventory_load_config.csv:<br>CSV_FILENAME does not exist<br>CSV_FILENAME is misspelled<br>A value other than 0 or 1 is in EXEC_BOOL<br>A STG_TABLE_NAME is misspelled |
| The script a.generate_inventory.bat fails and you see the command window. | In inventory_load_config.csv, a blank CSV_FILENAME value exists. |
| Report does not display properly (The gridlines are missing on the tables). | A browser other than Internet Explorer is being used to display the reports because the *.html file type is associated with some other browser. |

# Populating inventory staging tables

In order to use the Actuals Loader, you must first populate the staging tables. These staging tables contain all of the data that will be used by Actuals Loader for validation and importing.

* Denotes columns that are mandatory.

**Denotes columns that are conditionally mandatory.

## C_RI_INVENTORY (Root inventory)

The C_RI_INVENTORY table is used to load all aircrafts, assemblies and uninstalled inventory items.

| C_RI_INVENTORY | |
|---|---|
| SERIAL_NO_OEM | VARCHAR2(40) |
| PART_NO_OEM | VARCHAR2(40) |
| INV_CLASS_CD | VARCHAR2(8) |
| MANUFACT_CD | VARCHAR2(16) |
| AUTHORITY_CD | VARCHAR2(8) |
| LOC_CD | VARCHAR2(2000) |
| VENDOR_CD | VARCHAR2(16) |
| INV_COND_CD | VARCHAR2(8) |
| OWNER_CD | VARCHAR2(16) |
| ICN_NO_SDESC | VARCHAR2(80) |
| LOT_OEM_TAG | VARCHAR2(80) |
| MANUFACT_DT | DATE |
| PO_REF_SDESC | VARCHAR2(80) |
| RECEIVED_DT | DATE |
| NOTE | VARCHAR2(4000) |
| SHELF_EXPIRY_DT | DATE |
| BIN_QT | FLOAT |
| BATCH_NO_OEM | VARCHAR2(40) |
| INV_CAPABILITY_CD | VARCHAR2(8) |
| CARRIER_CD | VARCHAR2(8) |
| COUNTRY_CD | VARCHAR2(8) |
| AC_REG_CD | VARCHAR2(10) |
| AIRWORTH_CD | VARCHAR2(10) |
| FIN_NO_CD | VARCHAR2(40) |
| LINE_NO_OEM | VARCHAR2(40) |
| VAR_NO_OEM | VARCHAR2(40) |
| REG_BODY_CD | VARCHAR2(8) |
| INV_OPER_CD | VARCHAR2(8) |
| PRIVATE_BOOL | NUMBER |
| APPL_EFF_CD | VARCHAR2(8) |
| MOD_STATUS_NOTE | VARCHAR2(200) |
| ASSMBL_CD | VARCHAR2(8) |
| ASSMBL_BOM_CD | VARCHAR2(50) |
| ISSUE_ACCOUNT_CD | VARCHAR2(80) |
| FORECAST_MODEL | VARCHAR2(80) |
| CUSTOM_BARCODE | VARCHAR2(80) |

## C_RI_INVENTORY - ACFT

Aircraft assemblies use the following column specification:

***SERIAL_NO_OEM**: The serial number of the aircraft assembly.

***PART_NO_OEM** - The part number of the aircraft assembly.

**INV_CLASS_CD** – Class of the inventory is "ACFT" for aircraft.

**MANUFACT_CD** - Manufacturer code of the aircraft assembly.

**AUTHORITY_CD** - Authority code for authority over this aircraft assembly.

***LOC_CD** - Location code of the new aircraft. If INV_COND_CD is scrap or archive, the loc_cd must be a scrap or archive location.

**VENDOR_CD** - Purchase vendor code for the aircraft assembly.

**INV_COND_CD** - The code for the current condition of the aircraft assembly being inducted. Must be a valid entry in the REF_INV_COND table.

**Note** If INV_COND_CD is left blank, the following value will be automatically applied:

- If the aircraft hierarchy is complete, then inv_cond_cd will be set to INSRV.
- Otherwise, the aircraft inv_cond_cd will be set to REPREQ.

**OWNER_CD** - Owner code of the aircraft assembly.

**ICN_NO_SDESC** - Inventory control number for the aircraft assembly. This can be used as a secondary serial number.

**LOT_OEM_TAG** - The lot number of the aircraft assembly.

**MANUFACT_DT** - The manufacture date of the aircraft assembly.

**PO_REF_SDESC** - Purchase order for the aircraft assembly.

**RECEIVED_DT** - The date when the aircraft assembly was received.

**NOTE** - Miscellaneous note for the aircraft assembly.

**INV_CAPABILITY_CD** - Aircraft capability for an aircraft assembly. Must be a valid entry in the REF_INV_CAPABILITY table.

**CARRIER_CD** - Carrier of the aircraft assembly.

**COUNTRY_CD** - Country of the aircraft assembly. Must be a valid entry in the REF_COUNTRY table.

***AC_REG_CD** - Aircraft registration code of the aircraft assembly.

**AIRWORTH_CD** - The regulatory body's Airworthiness designation code for the aircraft assembly.

**FIN_NO_CD** - Fleet Identification Number (FIN) of the aircraft assembly.

**LINE_NO_OEM** - Line number of the original equipment manufacturer of the aircraft assembly.

**VAR_NO_OEM** - Variable number from the original equipment manufacturer of the aircraft assembly.

**REG_BODY_CD** - Defines the regulatory body associated with the aircraft assembly.

***INV_OPER_CD** - Defines overall serviceability of the aircraft assembly. Must be a valid entry in the REF_INV_OPER table.

**PRIVATE_BOOL** - Set to true if aircraft assembly is private; false for commercial.

**APPL_EFF_CD** - Aircraft applicability code.

**MOD_STATUS_NOTE** - The modification status of the inventor.

***ISSUE_ACCOUNT_CD** - The account code for the issue of aircraft type inventory.

**FORECAST_MODEL** – Forecast model associated to the aircraft.

## C_RI_INVENTORY - ASSY

Non-aircraft assemblies use the following column specification:

***SERIAL_NO_OEM** - The serial number of the non-aircraft assembly.

***PART_NO_OEM** - The part number of the non-aircraft assembly.

**MANUFACT_CD** - Manufacturer code of the non-aircraft assembly.

**AUTHORITY_CD** - Person who has authority over this non-aircraft assembly.

***LOC_CD** - Location of the new non-aircraft assembly. Installed assemblies must match its parent LOC_CD. If INV_COND_CD is scrap or archive, the LOC_CD must be a scrap or archive location

**VENDOR_CD** - Purchase vendor code for the non-aircraft assembly.

**INV_COND_CD** - The code for the current condition of the aircraft assembly being inducted. Must be a valid entry in the REF_INV_COND table.

**Note** If INV_COND_CD is left blank, its value will be automatically applied in the following order:

1. If the assembly has a parent that is Archived or Scrapped, INV_COND_CD wll be set to either ARCHIVE or SCRAP to match the parent.
2. If the assembly is installed, INV_COND_CD wll be set to INSRV.
3. If the assembly is loose and the assembly hierarchy is complete, INV_COND_CD wll be set to RFI.
4. If the assembly is loose and the assembly hierarchy is not complete, INV_COND_CD wll be set to REPREQ.

**OWNER_CD** – The owner code of the non-aircraft assembly.

**ICN_NO_SDESC** – The inventory control number for the non-aircraft assembly. This can be used as a secondary serial number.

**LOT_OEM_TAG** - The lot number of the non-aircraft assembly.

**MANUFACT_DT** - The manufacture date of the non-aircraft assembly.

**PO_REF_SDESC** - Purchase order for the non-aircraft assembly.

**RECEIVED_DT** - The date when the non-aircraft assembly was received.

**NOTE** - Miscellaneous note for the non-aircraft assembly.

**CARRIER_CD** - Carrier of the Engine or APU (ORG_CARRIER). This is required if Maintenance Programs are used in CAMO or if operator applicability will be used on the Engine or APU.

**APPL_EFF_CD** – The non-aircraft assembly applicability code.

**MOD_STATUS_NOTE** - The modification status of the inventory.

**CUSTOM_BARCODE** - The barcode for the inventory item. If this is left blank, then a barcode will be generated automatically.

## C_RI_INVENTORY - TRK

Uninstalled tracked items use the following column specification:

***SERIAL_NO_OEM** - The serial number of the uninstalled tracked inventory item.

***PART_NO_OEM** - The part number of the uninstalled tracked inventory item.

**MANUFACT_CD** - Manufacturer code of the uninstalled tracked inventory item.

**AUTHORITY_CD** - Person who has authority over this piece of uninstalled tracked inventory.

***LOC_CD** - Location of the new uninstalled tracked inventory item. If INV_COND_CD is scrap or archive, the loc_cd must be a scrap or archive location.

**VENDOR_CD** - Purchase vendor code for the uninstalled tracked inventory item.

**INV_COND_CD** - Current condition code of the uninstalled tracked inventory item being inducted. Must be a valid entry in the REF_INV_COND table. If this is left blank, then the condition will be set to RFI.

**OWNER_CD** – The owner code of the uninstalled tracked inventory item.

**ICN_NO_SDESC** – The inventory control number for the uninstalled tracked inventory item. Essentially this can be used as a secondary serial number.

**LOT_OEM_TAG** - The lot number of the uninstalled tracked inventory item.

**MANUFACT_DT** - The manufacture date of the uninstalled tracked inventory item.

**PO_REF_SDESC** – The purchase order for the uninstalled tracked inventory item.

**RECEIVED_DT** - The date when the uninstalled tracked inventory item was received.

**NOTE** - Miscellaneous note for the uninstalled tracked inventory item.

**SHELF_EXPIRY_DT** - Specifies shelf life expiry date of the uninstalled tracked inventory item.

**MOD_STATUS_NOTE** - The modification status of the inventory.

**ASSMBL_CD** - Specifies the assembly context for the component. This is required if the tracked item will have sub-components.

**ASSMBL_BOM_CD** - Specifies the configuration slot for the component. This is required if the tracked item will have sub-components.

**CUSTOM_BARCODE** - The barcode for the inventory item. If this is left blank, then a barcode will be generated automatically. quire

## C_RI_INVENTORY - SER

Uninstalled serialized items use the following column specification:

***SERIAL_NO_OEM** - The serial number of the installed untracked serialized inventory item.

***PART_NO_OEM** - The part number of the uninstalled untracked serialized inventory item.

**MANUFACT_CD** - Manufacturer code of the uninstalled, untracked serialized inventory item.

**AUTHORITY_CD** - Person who has authority over this piece of uninstalled untracked serialized inventory item.

***LOC_CD** - Location of the new uninstalled untracked serialized inventory item. If INV_COND_CD is scrap or archive, the loc_cd must be a scrap or archive location.

**VENDOR_CD** - Purchase vendor code for the uninstalled untracked serialized inventory item.

**INV_COND_CD** - Current condition code of the uninstalled untracked serialized inventory item being inducted. Must be a valid entry in the REF_INV_COND table.

> **Note** IIf INV_COND_CD is left blank, a default value of RFI will be applied.

**OWNER_CD** – The owner code of the uninstalled untracked serialized inventory item.

**ICN_NO_SDESC** – The inventory control number for the uninstalled untracked serialized inventory item. This can be used as a secondary serial number.

**LOT_OEM_TAG** - The lot number of the uninstalled untracked serialized inventory item.

**MANUFACT_DT** - The manufacture date of the uninstalled untracked serialized inventory item.

**PO_REF_SDESC** - Purchase order for the uninstalled untracked serialized inventory item.

**RECEIVED_DT** - The date when the uninstalled untracked serialized inventory item was received.

**NOTE** - Miscellaneous note for the uninstalled untracked serialized inventory item.

**SHELF_EXPIRY_DATE** - Specifies shelf-life expiry date of the uninstalled untracked serialized inventory item.

**MOD_STATUS_NOTE** - The modification status of the inventory.

**CUSTOM_BARCODE** - The barcode for the inventory item. If this is left blank, then a barcode will be generated automatically.

## C_RI_INVENTORY - BATCH

Uninstalled batch-controlled inventory items use the following column specification:

***SERIAL_NO_OEM** - The batch number for the batch controlled inventory item.

***PART_NO_OEM** - The part number of the untracked batch controlled inventory item.

**MANUFACT_CD** - Manufacturer code of the uninstalled, untracked batch controlled inventory item.

**AUTHORITY_CD** - Person who has authority over this piece of uninstalled, untracked batch controlled inventory.

***LOC_CD** - Location of the new untracked batch controlled inventory item. If INV_COND_CD is scrap or archive, the loc_cd must be a scrap or archive location.

**VENDOR_CD** - Purchase vendor code for the uninstalled, untracked batch controlled inventory item.

**INV_COND_CD** - Current condition code of the uninstalled, untracked batch controlled inventory item being inducted. Must be a valid entry in the REF_INV_COND table. If this is left blank, the inventory condition will be set to RFI.

**OWNER_CD** – The owner code of the uninstalled, untracked batch controlled inventory item.

**ICN_NO_SDESC** – The inventory control number for the uninstalled, untracked batch controlled inventory item. This can be used as a secondary serial number.

**LOT_OEM_TAG** - The lot number of the uninstalled, untracked batch controlled inventory item.

**MANUFACT_DT** - The manufacture date of the uninstalled, untracked batch controlled inventory item.

**PO_REF_SDESC** - Purchase order for the untracked batch controlled inventory item.

**RECEIVED_DT** - The date when the uninstalled, untracked batch controlled inventory item was received.

**NOTE** - Miscellaneous note for the uninstalled, untracked batch controlled inventory item.

**SHELF_EXPIRY_DATE** - Specifies shelf-life expiry date of the uninstalled, untracked batch controlled inventory item.

***BIN_QT** - The number of physical items stored within this batch of uninstalled, untracked batch control items.

**BATCH_NO_OEM** - The batch number of the uninstalled, untracked, batch controlled inventory.

**MOD_STATUS_NOTE** - The modification status of the inventory.

**CUSTOM_BARCODE** - The barcode for the inventory item. If this is left blank, then a barcode will be generated automatically.

### C_RI_INVENTORY - KIT

Kit Inventory items use the following column specification:

***SERIAL_NO_OEM** - The serial number of the kit inventory item.

***PART_NO_OEM** - The part number of the kit inventory item.

**MANUFACT_CD** - Manufacturer code of the kit inventory item.

**AUTHORITY_CD** - Person who has authority over this kit inventory item.

***LOC_CD** - Location of the new kit inventory item. If INV_COND_CD is scrap or archive, the loc_cd must be a scrap or archive location.

**VENDOR_CD** - Purchase vendor code for the kit inventory item.

**INV_COND_CD** - Current condition code of the kit inventory item being inducted. For kits, They cannot be set to INSRV, INREP or REPREQ. The default value, if INV_COND_CD is left blank will be RFI

**OWNER_CD** – The owner code of the kit inventory item.

**ICN_NO_SDESC** – The inventory control number for the kit inventory item. This can be used as a secondary serial number.

**LOT_OEM_TAG** - The lot number of the kit inventory item.

**MANUFACT_DT** - The manufacture date of the kit inventory item.

**PO_REF_SDESC** - Purchase order for the kit inventory item.

**RECEIVED_DT** - The date when the kit inventory item was received.

**NOTE** - Miscellaneous note for the kit inventory item.

**MOD_STATUS_NOTE** - The modification status of the inventory.

**CUSTOM_BARCODE** - The barcode for the inventory item. If this is left blank, then a barcode will be generated automatically.

## C_RI_INVENTORY_SUB (Sub Inventory)

**About this task**

The C_RI_INVENTORY_SUB staging table is used to load tracked and serialized sub-components that are currently installed. The parent item under which they are installed must be loaded using the C_RI_INVENTORY staging table.

Installed assemblies cannot be loaded using the C_RI_INVENTORY_SUB table. Instead, they must be loaded as loose inventory using the C_RI_INVENTORY staging table and then attached to their parent aircraft using the C_RI_ATTACH staging table.

| C_RI_INVENTORY_SUB | |
|---|---|
| PARENT_SERIAL_NO_OEM | VARCHAR2(40) |
| PARENT_PART_NO_OEM | VARCHAR2(40) |
| PARENT_MANUFACT_CD | VARCHAR2(16) |
| BOM_PART_CD | VARCHAR2(50) |
| EQP_POS_CD | VARCHAR2(200) |
| SERIAL_NO_OEM | VARCHAR2(40) |
| PART_NO_OEM | VARCHAR2(40) |
| MANUFACT_CD | VARCHAR2(16) |
| INV_CLASS_CD | VARCHAR2(8) |
| AUTHORITY_CD | VARCHAR2(8) |
| VENDOR_CD | VARCHAR2(16) |
| OWNER_CD | VARCHAR2(16) |
| ICN_NO_SDESC | VARCHAR2(80) |
| LOT_OEM_TAG | VARCHAR2(80) |
| MANUFACT_DT | DATE |
| PO_REF_SDESC | VARCHAR2(80) |
| RECEIVED_DT | DATE |
| INSTALL_DT | DATE |
| NOTE | VARCHAR2(4000) |
| SHELF_EXPIRY_DT | DATE |
| MOD_STATUS_NOTE | VARCHAR2(200) |
| CUSTOM_BARCODE | VARCHAR2(80) |

## C_RI_INVENTORY_SUB - TRK

Installed tracked items use the following column specifications:

***PARENT_SERIAL_NO_OEM** - The serial number of the parent component onto which this item is installed. This parent item must be listed in C_RI_INVENTORY.

***PARENT_PART_NO_OEM** - The part number of the parent component onto which this item is installed. This parent item must be listed in C_RI_INVENTORY.

***PARENT_MANUFACT_CD** - The manufacturer of the parent component onto which this item is installed. This parent item must be listed in C_RI_INVENTORY.

****BOM_PART_CD** - This is the Part Group Code of the configuration slot where this assembly is installed.

****EQP_POS_CD** - This is the Position Code within the configuration slot where this assembly is installed.

***SERIAL_NO_OEM** - The serial number of the installed inventory item.

***PART_NO_OEM** - Part number of the installed inventory item.

**MANUFACT_CD** - Manufacturer of the installed inventory item.

**INV_CLASS_CD** - Inventory class code.

**AUTHORITY_CD** - Person who has authority over this piece of installed inventory.

**VENDOR_CD** - Purchase vendor of this installed inventory item.

**OWNER_CD** - The owner of the installed inventory item.

**ICN_NO_SDESC** - The inventory control number for the installed inventory item. This can be used as a secondary serial number.

**LOT_OEM_TAG** - The lot number of the installed inventory item, if applicable.

**MANUFACT_DT** - Manufacture date of the installed inventory item.

**PO_REF_SDESC** - Purchase order for the installed inventory item.

**RECEIVED_DT** - The date when the installed item was received.

**INSTALL_DT** - The date of installation of the installed inventory item.

**NOTE** - Any miscellaneous note for the installed inventory item.

**SHELF_EXPIRY_DT** - Specifies shelf life expiry date of the installed inventory item.

**MOD_STATUS_NOTE** - The modification status of the inventory.

**CUSTOM_BARCODE** - The barcode for the inventory item. If this is left blank, then a barcode will be generated automatically.

## C_RI_INVENTORY_SUB - SER

Installed serialized items use the following column specifications:

***PARENT_SERIAL_NO_OEM** - The serial number of the parent component onto which this item is installed. This parent item must be listed in C_RI_INVENTORY.

***PARENT_PART_NO_OEM** - The part number of the parent component onto which this item is installed. This parent item must be listed in C_RI_INVENTORY.

***PARENT_MANUFACT_CD** - The manufacturer of the parent component onto which this item is installed. This parent item must be listed in C_RI_INVENTORY.

***SERIAL_NO_OEM** - The serial number of the installed inventory item.

***PART_NO_OEM** - Part number of the installed inventory item.

**MANUFACT_CD** - Manufacturer of the installed inventory item.

**INV_CLASS_CD** - Inventory class code.

**AUTHORITY_CD** - Person who has authority over this piece of installed inventory.

**VENDOR_CD** - Purchase vendor of this installed inventory item.

**OWNER_CD** - The owner of the installed inventory item.

**ICN_NO_SDESC** - The inventory control number for the installed inventory item. This can be used as a secondary serial number.

**LOT_OEM_TAG** - The lot number of the installed inventory item, if applicable.

**MANUFACT_DT** - Manufacture date of the installed inventory item.

**PO_REF_SDESC** - Purchase order for the installed inventory item.

**RECEIVED_DT** - The date when the installed item was received.

**INSTALL_DT** - The date of installation of the installed inventory item.

**NOTE** - Any miscellaneous note for the installed inventory item.

**SHELF_EXPIRY_DT** - Specifies shelf life expiry date of the installed inventory item.

**MOD_STATUS_NOTE** - The modification status of the inventory.

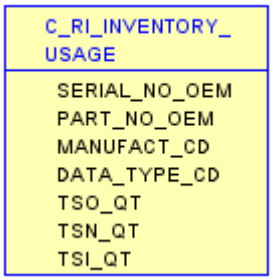
**CUSTOM_BARCODE** - The barcode for the inventory item. If this is left blank, then a barcode will be generated automatically.

**Note** The BOM_PART_CD field is not used for SER sub-components. If it is filled in, it will be ignored. This also means that CAMO does not check applicability rules for installed SER sub-components.

# C_RI_INVENTORY_USAGE (Inventory Usage)

### About this task

The C_RI_INVENTORY_USAGE staging table is used to load the current usage counts for every inventory item in C_RI_INVENTORY and C_RI_INVENTORY_SUB.




Inventory usage records use the following column specifications:

***SERIAL_NO_OEM** - Specifies the serial number of the applicable inventory item.

***PART_NO_OEM** - Specifies the part no of the applicable inventory item.

**MANUFACT_CD** - Specifies the manufacturer of the applicable inventory item.

***DATA_TYPE_CD** - Specifies data type of the usage parameter. Must be a valid entry in the MIM_DATA_TYPE table with DOMAIN_TYPE_CD = US.

**TSO_QT** - Time Since Overhaul value of the usage parameter. Default value is 0.

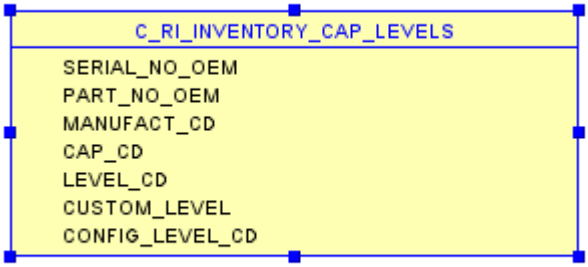
**TSN_QT** - Time Since New value of the usage parameter. Default value is 0.

**TSI_QT** - Time Since Install value of the usage parameter. Default value is 0.

## C_RI_INVENTORY_CAP_LEVELS

**About this task**

The C_RI_INVENTORY_CAP_LEVELS table is used to load the capability information for aircraft in C_RI_INVENTORY.




Inventory capability levels use the following column specifications:

***SERIAL_NO_OEM** - Specifies the serial number of the aircraft.

***PART_NO_OEM** - Specifies the part number of the aircraft.

**MANUFACT_CD** - Specifies the manufacturer of the aircraft.

**CAP_CD** - Specifies the capability for the aircraft.

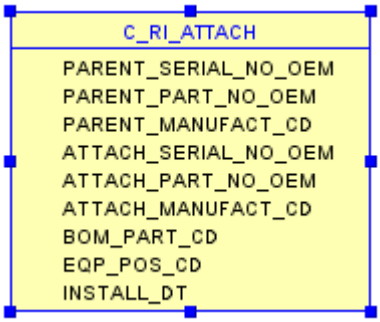
**LEVEL_CD** – Specifies the current capability level for the aircraft.

**CUSTOM_LEVEL** – Specifies a custom current capability level for the aircraft. Applicable to seat count only.

**CONFIG_LEVEL_CD** – Specifies the configured capability level for the aircraft.

## C_RI_ATTACH (Inventory attachments)

The C_RI_ATTACH staging table is used to attach a loose assembly onto a parent aircraft. Since the C_RI_INVENTORY_SUB staging table is not able to load installed assemblies, all installed assemblies must be loaded through C_RI_INVENTORY and then attached to the parent aircraft using C_RI_ATTACH.




Inventory attachment records use the following column specifications:

***PARENT_SERIAL_NO_OEM** - Specifies the OEM Serial No of the parent inventory item (aircraft).

***PARENT_PART_NO_OEM** - Specifies the Part No of the parent inventory item (aircraft).

**PARENT_MANUFACT_CD** - Specifies the Manufacturer of the parent inventory item (aircraft).

***ATTACH_SERIAL_NO_OEM** - Specifies the OEM Serial No of the installed assembly.

***ATTACH_PART_NO_OEM** - Specifies the Part No of the installed assembly.

**ATTACH_MANUFACT_CD** - Specifies the Manufacturer of the installed assembly.

**BOM_PART_CD** - This is the Part Group Code of the configuration slot where this assembly is installed

**EQP_POS_CD** - This the Position Code within the configuration slot where this assembly is installed

* **INSTALL_DT** - The date of the installation of the child assembly onto the parent assembly.

## Inventory data processing tables

The data loaded in inventory staging tables is used by the following processing tables:

- al_proc_inventory
- al_proc_inventory_sub
- al_proc_inventory_usage
- al_proc_attach
- al_proc_inventory_cap_levels

## Validating inventory data

Once the staging tables have been populated, you must validate the data and check for any problems or errors. Populating and validating is typically an iterative process; you populate the

staging tables, validate the data, resolve the errors by altering the data in the staging tables, and then validate again. You repeat the process until the validation succeeds.

You can validate inventory data in two ways:

- Using .bat scripts
- Using packages

## Validating inventory data using .bat file

You can validate inventory data by running a .bat script file.

**About this task**

To validate inventory data using a .bat script, do the following:

**Steps**

1. Validate the data in the staging tables by running the following script file. Depending on the amount of data you have to import, this step might take several minutes to complete. `<installation_folder>\dataload\script_Actuals\inventory\b.validate_inventory.bat`
2. When the script completes, a browser window opens showing the results of the validation.

**Note** If you see a little popup window at the bottom of your browser window, click **Allow blocked content**. Otherwise some information will not be displayed.




The report summarizes the data from each staging table as follows:

- **Staging**: indicates the staging table name
- **Processed**: indicates how many records were loaded
- **Valid**: indicates how many loaded records were valid
- **Error**: indicates how many loaded records were invalid due to errors
- **Percentage**: indicates the percentage of valid records out of the total number of records

There is also a listing of all of the validation errors found in each staging table as follows. You can click any Error Code to see more information about which records contain this particular error. All errors must be resolved and corrected in the staging table before you can import the data. If there are any warnings, they will show up in italics and do not need to be corrected before performing the import.

## Validating inventory data using packages

You can use the package named MX_AL_CTRLLER_PKG to validate inventory data.

**Package name:**

MX_AL_CTRLLER_PKG

**Procedure name:**

EXECUTE_INV_VALIDATION

**Input arguments:**

- aiv_complete_assy_bool: Value can be "Full" create parts for every position on the baseline whereas "Not" will leave position empty if the part was not specified in the import.
- ain_gather_stats_bool: Turn gather stats on or off.

**Output arguments:**

- aon_retcode:
  - 1: This error message means that the validate procedure executed successfully without any unexpected errors. It doesn't mean that data is valid or not.
  - -1: Error occurred while executing the package.
- aov_retmsg: Outcome description

## Error handling for inventory data

If there are any errors encountered while validating the inventory data, appropriate error codes will be added to the following table:

- al_proc_inventory_error

You can view the definitions for the error messages in the following table:

- dl_ref_message

Any mandatory subcomponents that are missing from your inventory will be listed in the table al_missing_inventory. Inventory loading can still succeed without these missing subcomponents. The missing subcomponents can be automatically created by setting complete.assy.bool=FULL in the runtime.properties file, or with the input argument aiv_complete_assy_bool = FULL when running the inventory import procedure.

# Importing inventory data

Once the data has been validated, it can be imported into CAMO.

You can import inventory data in two ways:

- Using .bat scripts
- Using packages

## Importing inventory data using .bat file

You can import inventory data by running a .bat script file.

**About this task**

Once the inventory data has been validated, it can be imported into CAMO.

**Steps**

1. Import the data from the staging tables by running the following script file: `<installation_folder>\dataload>\script_Actuals\inventory \c.import_inventory.bat`.
2. When the script completes, a browser opens an Actuals Loader Report displaying the results of the import. The report summarizes the data from each staging table as follows:
   - **Staging:** indicates the staging table name
   - **Processed:** indicates how many records were loaded
   - **Imported:** indicates how many loaded records were imported
   - **Error:** indicates how many loaded records were not able to be imported due to errors
   - **Percentage:** indicates the percentage of imported records out of the total number of records

## Importing inventory data using packages

You can use the package named MX_AL_CTRLLER_PKG to validate inventory data.

**Package name:**

MX_AL_CTRLLER_PKG

**Procedure name:**

EXECUTE_INV_IMPORT

**Input arguments:**

- aiv_complete_assy_bool: Value can be "Full" create parts for every position on the baseline whereas "Not" will leave position empty if the part was not specified in the import.
- ain_parallelism_degree: Number of dedicated processors.
- ain_chunk_size: Number of records processed by one processor.
- ain_gather_stats_bool: Turn gather stats on or off.

**Output arguments:**

- aon_retcode:
  - 1: This error message means that the import procedure executed successfully without any unexpected errors.
  - -1: Error occurred while executing the package.
- aov_retmsg: Outcome description

# Finalizing inventory data

**About this task**

Once all of the tasks, faults and work packages have been created against your imported inventory, there are a few things that need to be done to finalize the loaded inventory. These are activities such as:

- Converting unknown sub-components' serial number "AL-XXX-{unique identifier}" to serial number "XXX"

## Finalizing inventory data using .bat file

You can finalize inventory data by running a .bat script file.

To finalize inventory data using a .bat file, do the following:

- Import the data from the staging tables into CAMO by running the following script file:

  ```
  <installation_folder>\dataload\script_Actuals\inventory
  \d.finalize_inventory.bat
  ```
- When the script completes, the command window will show "Finalize successful..." The inventory has now been fully loaded and ready for use.

## Finalizing inventory data using procedure

A procedure named AL_FINALIZE_INVENTORY is called directly to finalize importing the inventory data. It accepts no input or output parameters.

**Procedure name:**

AL_FINALIZE_INVENTORY

# 4

# Loading historical usage

After loading the data for inventory, you can load the data for assembly historical usage.

## Loading historical usage using .csv files

**About this task**

In order to use the Actuals Loader, you must populate the staging tables described in the next section. Instead of using SQL to fill in these staging tables, Actuals Loader provides the option to fill in .csv files and then load those .csv files into the staging tables.

**Steps**

1. Fill out your `AL_HISTORICAL_USAGE.csv` files following these example templates found in `<installation_folder>\dataload>\script_Actuals\usage\data\AL_HISTORICAL_USAGE_sample.csv`
2. Into the same directory, copy your completed `AL_HISTORICAL_USAGE.csv` files.
3. Update columns A, C, D and E of the file `<installation_folder>\dataload\script_Actuals\usage\data\usage_load_config.csv` with the details of your `AL_HISTORICAL_USAGE.csv` files.

   This file contains configuration information for the usage loading script. The figure below shows an example of a completed usage_load_config.csv file.

| CSV_FILENAME | STG_TABLE_NAME | NUMBER_TITLE_ROWS | DATE_FORMAT | EXEC_BOOL |
|---|---|---|---|---|
| AL_HISTORICAL_USAGE_sample.csv | AL_HISTORICAL_USAGE | 1 | MM/DD/YYYY HH24:MI | 1 |

   - CSV_FILENAME (column A): List AL_HISTORICAL_USAGE.csv file that you would like to load into the staging tables. Make sure that none of these are blank, otherwise the loading script will fail.
   - STG_TABLE_NAME (column B): The corresponding Oracle staging table name. Do not change these values.

- NUMBER_TITLE_ROWS (column C): List the number of rows used for titles in your file from column A. If the data in your AL_HISTORICAL_USAGE.csv data file does not start at the 2nd row, increment this number to show how many title rows to skip before the inventory loading script looks for your data.
- DATE_FORMAT (column D): List the format of the dates (COLLECTION_DT) used in your file from column A. All dates must have consistent format within each AL_HISTORICAL_USAGE.csv file, otherwise the validation will have errors.
- EXEC_BOOL (column E): This is a binary value. If the value is a 1, the data loading script will load the file's data into the staging tables. If the value is a 0, the data loading script will skip over this file and will not load it.

4. Load the inventory files into the staging tables by running the script file `<installation_folder>\dataload\script_Actuals\usage\a.load_usage.bat`

When the script completes, it displays a command window showing a status of the loading process. Usually it says 'BUILD SUCCESSFUL' for successful import and 'BUILD FAILED' for if it load failed displayed at the end of the command window. Check staging table AL_HISTORICAL_USAGE to see if data has been loaded.

# Populating historical usage staging tables

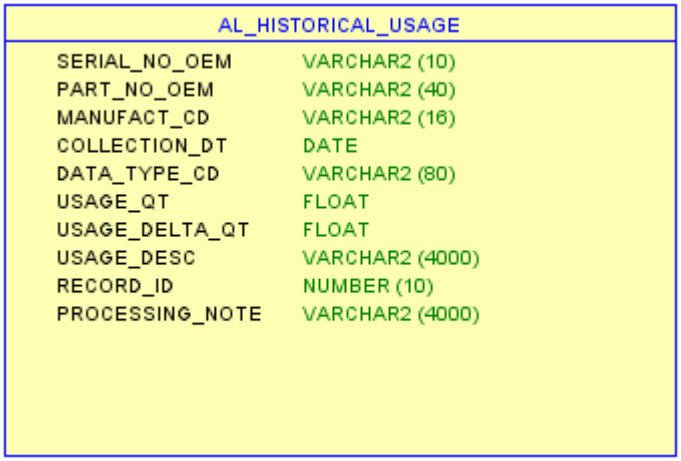
In order to use the Actuals Loader, you must first populate the staging tables. These staging tables contain all of the data that will be used by Actuals Loader for validation and importing

* Denotes columns that are mandatory.

**Denotes columns that are conditionally mandatory.

## AL_HISTORICAL_USAGE

The AL_HISTORICAL_USAGE table is used to load all historical data for assemblies.



*AL_HISTORICAL_USAGE table columns*

| Column | Description |
|---|---|
| *SERIAL_NO_OEM | The original equipment manufacturer's serial number |
| *PART_NO_OEM | Identifies the Part Number of the ACFT or ASSY that will accrue usage |
| MANUFACT_CD | Identifies the Manufacturer of the equipment that will accrue usage |
| *COLLECTION_DT | Identifies the date that the Usage Record was collected. The time (hours and minutes) is provided from the departure time |
| *DATA_TYPE_CD | A user assigned code that helps identify the parameter |
| *USAGE_QT | Identifies the total usage since new on the item |
| *USAGE_DELTA_QT | Usage increment from the previous record |
| USAGE_DESC | Detailed description of the usage record |
| RECORD_ID | Identifies row uniquely. Sequence-generated unique Id. Users should leave this field blank |
| PROCESSING_NOTE | Optional processing note |

## Historical usage data processing tables

The data loaded in usage staging tables is used by the following processing tables:

- al_proc_hist_usage

## Validating historical usage data

Once the staging tables have been populated, you must validate the data and check for any problems or errors. Populating and validating is typically an iterative process; you populate the staging tables, validate the data, resolve the errors by altering the data in the staging tables, and then validate again. You repeat the process until the validation succeeds.

You can validate historical usage data in two ways:

- Using .bat scripts

- Using packages

## Validating historical usage data using .bat file

**Steps**

1. Validate the data in the staging tables by running the file: `<installation_folder>\dataload\script_Actuals\odf\b.validate_usage.bat`

   Depending on the amount of data you have to import, this step might take several minutes to complete.

   When the script completes, it will say "BUILD SUCCESSFUL". This just means that validation has been successfully completed. In case the process failed, it will display error which can further diagnosed upon careful analysis.
2. To see the validation results, query the AL_PROC_HIST_USAGE table (for the usage header info) and the AL_PROC_RESULT table. Inner Join the above tables by RECORD_ID and select all key values from AL_PROC_HIST_USAGE.

   Sample query:

```
SELECT result_cd, al_proc_hist_usage.* FROM al_proc_hist_usage
INNER JOIN al_proc_result
   ON al_proc_hist_usage.record_id = al_proc_result.record_id;
```

## Validating historical usage data using packages

**Package name:**

AL_HISTORICAL_USAGE_IMPORT

**Procedure name:**

VALIDATE

**Input arguments:**

- ain_parallelism_degree: Number of dedicated processors.
- ain_chunk_size: Number of records processed by one processor.
- ain_gather_stats_bool: Turn gather stats on or off.

**Output arguments:**

- aon_retcode:
  - 1: This error message means that validate procedure executed successfully without any unexpected errors. It doesn't mean that data is valid or not.
  - -1: Error occurred while executing the package.
- aov_retmsg: Outcome description.

## Error handling for usage data

If there are any errors encountered while validating the usage data, appropriate error codes will be added to the following table:

- al_proc_result

You can view the definitions for the error messages in the following table:

- dl_ref_message

# Importing historical usage data

Once the data has been validated, it can be imported into CAMO.

You can import historical usage data in two ways:

- Using .bat scripts
- Using packages

## Importing historical usage data using .bat file

**Steps**

Run the script file: `<installation_folder>\dataload\script_Actuals\usage\c.import_usage.bat`

When the script completes, it displays a command window showing a status of the loading process. Usually it says ‘BUILD SUCCESSFUL’ for successful import and ‘BUILD FAILED’ for if it import failed displayed at the end of the command window. Check staging table AL_HISTORICAL_USAGE to see if data has been loaded.

## Importing historical usage data using packages

**Package name:**

AL_HISTORICAL_USAGE_IMPORT

**Procedure name:**

IMPORT

**Input arguments:**

- ain_parallelism_degree: Number of dedicated processors.
- ain_chunk_size: Number of records processed by one processor.
- ain_gather_stats_bool: Turn gather stats on or off.

**Output arguments:**

- aon_retcode:

  - 1: This error message means that import procedure executed successfully without any unexpected errors.
  - -1: Error occurred while executing the package.
- aov_retmsg: Outcome description

# 5

# Loading tasks

After loading the data for inventory and historical usage, you can load the data for maintenance tasks and reference documents.

The Actuals Loader collects historic task information for the purposes of initializing the maintenance history and the next due maintenance in CAMO. The task staging tables allow a user to specify the date of completion and the usage values at completion for the historic task. When a recurring task is loaded, the next due (i.e. Active) task will also be created.

The task table allows historic tasks based on task definitions to be loaded into CAMO. Historic tasks can be created for Requirement, Block and reference document task definitions defined on a configuration slot (root, system, or tracked config slot) or on the part number (Tracked, Serialized). The tasks will be loaded as "Completed" status and the date of completion and usage at completion will be indicated on the task.

Once the historic tasks are loaded, Actuals Loader will then generate the next due (i.e. Active) tasks for the task definition. Actuals Loader will use the completion information from the last done task, the task definition’s scheduling rules, and the scheduling engine to determine the correct due date and usage for the next due tasks. However, if the scheduling engine is not generating the desired due date and usage, you are able to override the deadline on the next due task, and provide custom start values or custom due values. If a task is no longer being performed, it should be loaded with a terminated date, the prevent the active task from being created.

If the actuals loader will be used in several iterative loads, all of the historic last done tasks for an asset should be loaded into the database before enabling synchronization on that asset.

# Loading task .csv files

**About this task**

In order to use the Actuals Loader, you must populate the staging tables described in the next section. Instead of using SQL to fill in these staging tables, Actuals Loader provides the option to fill in CSV files and then load those CSV files into the staging tables.

**Steps**

1. Fill out your `C_RI_*.csv` files following these example templates found in the`<installation_folder>\dataload\script_Actuals\inventory\data` folder:
   - `C_RI_TASK_sample.csv`
   - `C_RI_TASK_SCHED_sample.csv`
2. In the same directory, save your completed files:
   - `C_RI_TASK.csv`
   - `C_RI_TASK_SCHED.csv`
3. Modify the file `<database_install>\dataload\dataload\script_Actuals\task\data\task_load_config.csv`.

   This file contains configuration information for the task loading script. Starting in the second row, edit your task_load_config.csv file parameters.

   > **Note** Do not remove the column headers in the first row.

   - CSV_FILENAME (column A): list each C_RI_*.csv file that you would like to load into the staging tables.
   - STG_TABLE_NAME (column B): the corresponding Oracle staging table name. Do not change these values.
   - NUMBER_TITLE_ROWS (column C): List the number of rows used for titles in your file from column A. If the data in your C_RI_*.csv data file does not start at the 2nd row, increment this number to show how many title rows to skip before the task loading script will look for your data.
   - DATE_FORMAT (column D): list the format of the dates (COMPLETION_DATE, DUE_DATE, etc.) used in your file from column A. All dates must have consistent format within each AL_*.csv file, otherwise the validation will have errors.
   - EXEC_BOOL (column E): This is a binary value. If the value is a 1, the data loading script will try to load into the staging tables. If the value is a 0, the data loading script will skip over this file and will not try to load it.
4. Load the task files into the staging tables by running the script file `<database_install>\dataloaddataload\script_Actuals\task\a.load_task.bat`

When the script completes, it displays a report containing the status of the loading process.

# Populating task staging tables

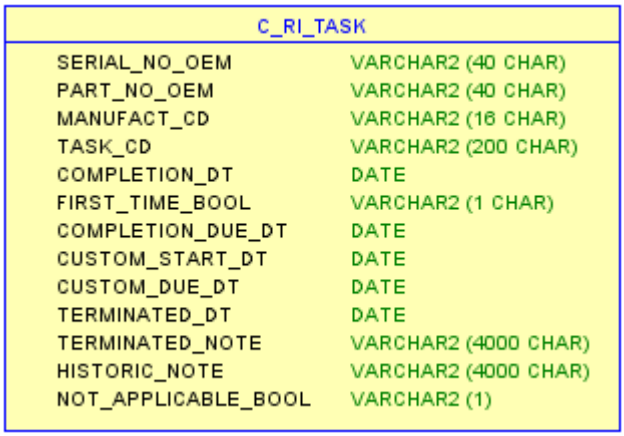
In order to use the Actuals Loader, you must first populate the staging tables. These staging tables contain all of the data that will be used by Actuals Loader for validation and importing.

* Denotes columns that are mandatory.

**Denotes columns that are conditionally mandatory.

## C_RI_TASK (task)

The C_RI_TASK staging table is used to identify all of the tasks that are to be loaded and their calendar information such as completion dates and due dates. Blocks and requirements can be loaded as historic, active, terminated or not applicable. Reference documents can be loaded as historic or not applicable.




*C_RI_TASK table columns*

| Column | Description |
|---|---|
| *SERIAL_NO_OEM | Serial number of the inventory against which the historic task was completed. When a task is on a system slot, the serial number is that of the parent assembly. |
| *PART_NO_OEM | Part number of the inventory against which the historic task was completed. When a task is on a system slot, the part number is that of the parent assembly. |

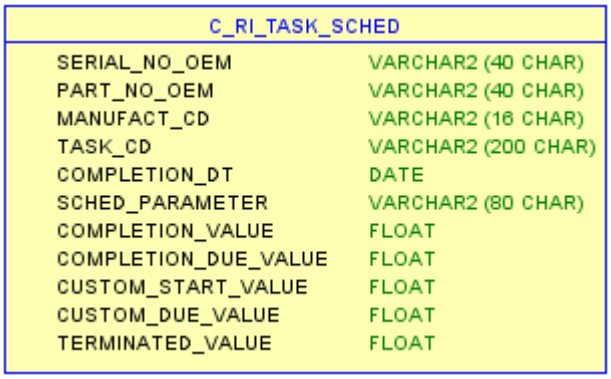
| Column | Description |
|---|---|
| *MANUFACT_CD | Manufacturer code of the inventory against which the historic task was completed. |
| *TASK_CD | Task definition of the historic task. |
| COMPLETION_DT | Completion date of the historic task. This field is used to specify historic completion of the task. This field is mandatory unless the task is being performed for the first time, in which case it should be left blank. |
| FIRST_TIME_BOOL | Boolean to indicate if the task is being done for the first time. Enter Y for Yes and N or null for No. |
| COMPLETION_DUE_DT | Due date of the historic task. This determines if the historic task was completed within the task definition schedule to plan window. |
| CUSTOM_START_DT | Custom start date of the active task. This will override the deadlines generated by the scheduling engine. |
| CUSTOM_DUE_DT | Custom due date of the active task. |
| TERMINATED_DT | The date when the task was terminated. Leave this empty if the task has not been terminated. |
| TERMINATED_NOTE | This is a Note that will be appended to the terminated task's audit trail. This audit trail is visible through the UI on the task's History tab. This field is optional. |
| HISTORIC_NOTE | This is a Note that will be appended to the historic task's audit trail. This audit trail is visible through the UI on the task's History tab. This field is optional. |
| NOT_APPLICABLE_BOOL | Boolean to indicate if the task is to be initialized as Not Applicable. |

**Note**

1. You cannot provide dates for both CUSTOM_START_DATE and CUSTOM_DUE_DATE; you may only provide one or the other.
2. If a task has been terminated, then you cannot provide dates for CUSTOM_START_DATE or CUSTOM_DUE_DATE.
3. You cannot provide a date for CUSTOM_DUE_DATE without providing a CUSTOM_DUE_VALUE in the C_RI_TASK_SCHED staging table for every scheduling parameter on the requirement.
4. You cannot provide a FIRST_TIME_BOOL value for more than one rows in a task. You must provide a value for a maximum of one row per task.
5. Reference documents can only be loaded as historic (COMPLETION_DT) or not applicable (NOT_APPLICABLE_BOOL). You cannot provide FIRST_TIME_BOOL or TERMINATED_DT for reference documents
6. You cannot provide the same COMPLETION_DT value multiple times for the same task. You must provide a different COMPLETION_DT value on each row of a given task.
7. You must associate every row in C_RI_TASK_SCHED to a parent row in C_RI_TASK.

## C_RI_TASK_SHED

The C_RI_TASK_SCHED table is used to load all of the usage-based information from tasks in the C_RI_TASK table.



*C_RI_TASK_SCHED table columns*

| Column | Description |
|---|---|
| *SERIAL_NO_OEM | Serial number of the inventory against which the historic task was completed. When a task is on a system slot, the serial number is that of the parent assembly. |
| *PART_NO_OEM | Part number of the inventory against which the historic task was completed. When a task is on a system slot, the part number is that of the parent assembly. |
| MANUFACT_CD | Manufacturer code of the inventory against which the historic task was completed. When a part exists in the system that is used by two or more manufacturers, the manufacturer code is mandatory. When a task is on a system slot, the manufacturer code is that of the parent assembly. |
| *TASK_CD | Task definition of the historic task. |
| COMPLETION_DT | Completion date of the historic task. This field is used to specify historic completion of the task. This field is mandatory unless the task is being performed for the first time, in which case it should be left blank. It should match the value specified in the C_RI_TASK table. |
| *SCHED_PARAMETER | The usage based parameter by which this task is scheduled, for example, HOURS or CYCLES. Calendar based parameters should not be specified in this field (or any following scheduling parameter field). Important: You must provide at least one usage parameter for every usage based task to be loaded properly. |
| *COMPLETION_VALUE | Inventory usage value at the completion of the historic task. Calendar based parameters should not be specified in this field (or any following completion value field). |

| Column | Description |
|---|---|
| COMPLETION_DUE_VALUE | Inventory usage when the task was last due. This determines if the historic task was completed within the task definition schedule to plan window. Calendar based parameters should not be specified in this field (or any following due value field). |
| CUSTOM_START_VALUE | Custom start value of the active task. This will override the deadlines generated by the scheduling engine. |
| CUSTOM_DUE_VALUE | Custom due value of the active task. |
| TERMINATED_VALUE | Inventory usage when the task was terminated. |

**Note**

1. You cannot provide a value for both CUSTOM_START_VALUE and CUSTOM_DUE_VALUE; you may only provide one or the other.
2. If the task is terminated, then you cannot provide values for CUSTOM_START_VALUE or CUSTOM_DUE_VALUE.
3. You cannot provide a value for CUSTOM_DUE_VALUE without providing custom due values for every other scheduling parameter on the requirement.
4. You cannot provide a value for CUSTOM_DUE_VALUE without providing a date for CUSTOM_DUE_DT in the C_RI_TASK staging table for the same requirement.

## Task data processing tables

The data loaded in task staging tables is used by the following processing tables:

- al_proc_tasks_biz_log
- al_proc_tasks_parameter
- al_proc_task_chain
- al_proc_task_chunk
- al_proc_task_imp_rpt
- al_proc_task_inv_cross
- al_proc_hist_task_parameter
- al_proc_hist_task
- al_proc_repl_tasks

- al_proc_repl_wkpg

# Validating task data

Once the staging tables have been populated, you must validate the data and check for any problems or errors. Populating and validating is typically an iterative process; you populate the staging tables, validate the data, resolve the errors by altering the data in the staging tables, and then validate again. You repeat the process until the validation succeeds.

You can validate task data in two ways:

- Using .bat scripts
- Using packages

## Validating task data using .bat file

In addition to the results for C_RI_TASK and C_RI_TASK_SCHED, the validation report displays a Missing Tasks row that displays the number of applicable tasks that are missing from the staging table. You can choose to load these tasks automatically by setting auto.create.task.bool=true in the runtime.propertiles file, however task loading will succeed without them.

**Steps**

1. Validate the data in the staging tables by running this script file: `<installation_folder>\dataload\script_Actuals\task\b.validate_task.bat`

   Depending on the amount of data you have to import, this step might take several minutes to complete.

   When the script completes, a browser window opens showing the results of the validation
2. If a popup window appears at the bottom of your browser, click **Allow blocked content**.

   If you do not click **Allow blocked content**, some information will not be displayed.
3. You may select an Error Code to see more information about which records contain the error.

   All errors must be resolved and corrected in the staging table before you can import the data.

   If there are any warnings, they will show up in italics and do not need to be corrected before importing.

The validation report summarizes the data from each staging table as follows:

- Staging: indicates the staging table name
- Processed: indicates how many records were loaded
- Valid: indicates how many loaded records were valid
- Error: indicates how many loaded records were invalid due to errors
- Percentage: indicates the percentage of valid records out of the total number of records

## Validating task data using packages

**Package name:**

MX_AL_CTRLLER_PKG

**Procedure name:**

EXECUTE_TASK_VALIDATION

**Input arguments:**

- ain_gather_stats_bool: Turn gather stats on or off.

**Output arguments:**

- aon_retcode:
  - 1: This error message means that validate procedure executed successfully without any unexpected errors. It doesn't mean that data is valid or not.
  - -1: Error occurred while executing the package.
- aov_retmsg: Outcome description

## The Missing Task Report

Any tasks that were not specified in C_RI_TASK, will be listed in the table al_missing_task. You can choose to load these tasks automatically by setting auto.create.task.bool=true in the runtime.properties file or with the input argument ain_auto_create_bool=1 in the import package, however task loading can succeed without them.

The tasks can be viewed by querying the al_missing_task table. A sample query is below.

```
SELECT
   CASE WHEN inv_inv.inv_class_cd = 'SYS'
        THEN assy_inv.serial_no_oem ELSE inv_inv.serial_no_oem END AS
 serial_no_oem,
   CASE WHEN inv_inv.inv_class_cd = 'SYS'
        THEN assy_part.part_no_oem ELSE eqp_part_no.part_no_oem END AS
 part_no_oem,
   CASE WHEN inv_inv.inv_class_cd = 'SYS'
        THEN assy_part.manufact_cd ELSE eqp_part_no.manufact_cd END AS
 manufact_cd,
   task_cd,
   owner_cd,
   task_name,
   ext_key_sdesc,
   task_class_cd,
   task_subclass_cd,
   task_task.assmbl_cd,
   task_bom.assmbl_bom_cd AS task_config_slot,
   acft_inv.serial_no_oem AS acft_serial_no_oem,
   inv_ac_reg.ac_reg_cd,
   acft_part.part_no_oem AS acft_part_no_oem,
   acft_part.manufact_cd AS acft_manufact_cd,
```

```
   assy_inv.serial_no_oem AS assy_serial_no_oem,
   assy_part.part_no_oem AS assy_part_no_oem,
   assy_part.manufact_cd AS assy_manufact_cd,
   inv_bom.assmbl_bom_cd AS inv_config_slot,
   eqp_assmbl_pos.eqp_pos_cd AS inv_position,
   CASE WHEN inv_inv.inv_class_cd = 'SYS'
        THEN assy_inv.barcode_sdesc ELSE inv_inv.barcode_sdesc END as
 inv_barcode
FROM
   al_missing_task
   INNER JOIN inv_inv ON
      inv_inv.inv_no_db_id = al_missing_task.inv_no_db_id AND
      inv_inv.inv_no_id = al_missing_task.inv_no_id
   INNER JOIN eqp_part_no ON
      eqp_part_no.part_no_db_id = inv_inv.part_no_db_id AND
      eqp_part_no.part_no_id = inv_inv.part_no_id
   INNER JOIN inv_owner ON
      inv_owner.owner_db_id = inv_inv.owner_db_id AND
      inv_owner.owner_id = inv_inv.owner_id
   LEFT OUTER JOIN eqp_assmbl_bom inv_bom ON
      inv_bom.assmbl_db_id = inv_inv.assmbl_db_id AND
      inv_bom.assmbl_cd = inv_inv.assmbl_cd AND
      inv_bom.assmbl_bom_id = inv_inv.assmbl_bom_id
   LEFT OUTER JOIN eqp_assmbl_pos ON
      eqp_assmbl_pos.assmbl_db_id = inv_inv.assmbl_db_id AND
      eqp_assmbl_pos.assmbl_cd = inv_inv.assmbl_cd AND
      eqp_assmbl_pos.assmbl_bom_id = inv_inv.assmbl_bom_id AND
      eqp_assmbl_pos.assmbl_pos_id = inv_inv.assmbl_pos_id
   LEFT OUTER JOIN inv_inv assy_inv ON
      assy_inv.inv_no_db_id = inv_inv.assmbl_inv_no_db_id AND
      assy_inv.inv_no_id = inv_inv.assmbl_inv_no_id
   LEFT OUTER JOIN eqp_part_no assy_part ON
      assy_part.part_no_db_id = assy_inv.part_no_db_id AND
      assy_part.part_no_id = assy_inv.part_no_id
   LEFT OUTER JOIN inv_inv acft_inv ON
      acft_inv.inv_no_db_id = inv_inv.h_inv_no_db_id AND
      acft_inv.inv_no_id = inv_inv.h_inv_no_id AND
      acft_inv.inv_class_cd = 'ACFT'
   LEFT OUTER JOIN eqp_part_no acft_part ON
      acft_part.part_no_db_id = acft_inv.part_no_db_id AND
      acft_part.part_no_id = acft_inv.part_no_id
   LEFT OUTER JOIN inv_ac_reg ON
      inv_ac_reg.inv_no_db_id = inv_inv.h_inv_no_db_id AND
      inv_ac_reg.inv_no_id = inv_inv.h_inv_no_id
   INNER JOIN task_task ON
      task_task.task_db_id = al_missing_task.task_db_id AND
      task_task.task_id = al_missing_task.task_id
   LEFT OUTER JOIN eqp_assmbl_bom task_bom ON
      task_bom.assmbl_db_id = task_task.assmbl_db_id AND
      task_bom.assmbl_cd = task_task.assmbl_cd AND
      task_bom.assmbl_bom_id = task_task.assmbl_bom_id;
```

## Error handling for task data

If there are any errors encountered while validating the task data, appropriate error codes will be added to the following table:

- al_proc_tasks_error

You can view the definitions for the error messages in the following table:

- dl_ref_message

# Importing task data

Once the data has been validated, it can be imported into CAMO.

You can import task data in two ways:

- Using .bat scripts
- Using packages

## Importing task data using .bat file

**Steps**

Import the data from the staging tables into CAMO by running the script file `<installation_folder>\dataload\script_Actuals\task\c.import_task.bat`

When the script completes, a browser opens an Actuals Loader Report displaying the results of the import. The report summarizes the data from each staging table as follows:

- **Staging**: indicates the staging table name.
- **Processed**: indicates how many records were loaded.
- **Imported**: indicates how many loaded records were imported.
- **Error**: indicates how many loaded records were not able to be imported due to errors.
- **Percentage**: indicates the percentage of imported records out of the total number of records.

## Importing task data using packages

**Package name:**

MX_AL_CTRLLER_PKG

**Procedure name:**

EXECUTE_TASK_IMPORT

**Input arguments:**

- ain_parallelism_degree: Number of dedicated processors.
- ain_chunk_size: Number of records processed by one processor.

- ain_gather_stats_bool: Turn gather stats on or off.
- ain_logging_bool: Turn logging on or off.
- ain_auto_create_bool: Enable or disable auto-creation of missing tasks

**Output arguments:**

- aon_retcode:
  - 1: This error message means that the import procedure executed successfully without any unexpected errors.
  - -1: Error occurred while executing the package.
- aov_retmsg: Outcome description

# 6

# Loading faults

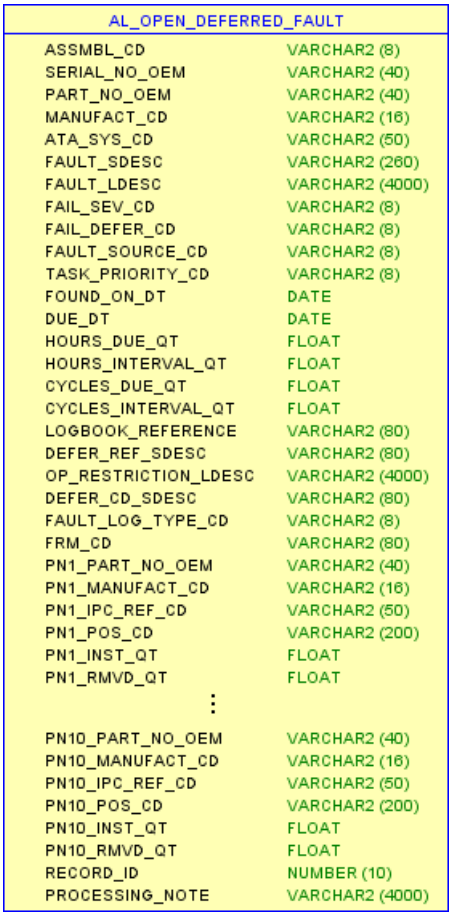
After loading the data for inventory, you can load the data for faults. The Actuals Loader allows you to load the list of open and deferred faults for aircraft and assemblies.

## Populating fault staging tables

In order to use the Actuals Loader, you must first populate the staging tables. These staging tables contain all of the data that will be used by Actuals Loader for validation and importing.

### AL_OPEN_DEFERRED_FAULT

The AL_OPEN_DEFERRED_FAULT table allows the entry of data applicable to Open Deferred Faults. The table lists all open deferred faults to be loaded in the database. The setup of this data creates deferred fault event records in the CAMO database.




*AL_OPEN_DEFERRED_FAULT table columns*

| Column | Description |
|---|---|
| ASSMBL_CD | This column is currently ignored by the Actuals Loader.<br><br>**Note** This column is deprecated and will be removed in a future release. |
| SERIAL_NO_OEM | The aircraft serial number that the fault is logged against. |
| [1]PART_NO_OEM | The aircraft part number that the fault is logged against. |
| MANUFACT_CD | The aircraft manufacturer that the fault is logged against. |

[1] Denotes columns that are mandatory

| Column | Description |
|---|---|
| ATA_SYS_CD | The ATA system code of the item that the fault is logged against. If the value for this field is NULL, the fault is open on the aircraft itself. |
| [1]FAULT_SDESC | The short description of the fault. |
| FAULT_LDESC | The long description of the fault. |
| [1]FAIL_SEV_CD | The code value indicating the severity of the fault. Must be a valid value in the REF_FAIL_SEV table. |
| FAIL_DEFER_CD | The code value indicating the reason for the deferral of the fault. Only valid for MEL type faults (REF_FAIL_SEV.SEV_TYPE_CD = 'MEL'). Must be a valid value in the REF_FAIL_DEFER table. |
| FAULT_SOURCE_CD | The code value indicating the source of the fault. Must be a valid entry in the REF_FAULT_SOURCE table. |
| TASK_PRIORITY_CD | The code value indicating the priority of the fault. Must be a valid entry in the REF_TASK_PRIORITY table. |
| [1]FOUND_ON_DT | The date when the fault was found. |
| DUE_DT | The due date of the fault. |
| HOURS_DUE_QT | The starting point to calculate the event deadline value in HOURS. |
| HOURS_INTERVAL_QT | The event interval value in HOURS. |
| CYCLES_DUE_QT | The starting point to calculate the event deadline value in CYCLES. |
| CYCLES_INTERVAL_QT | The event interval value in HOURS. |
| LOGBOOK_REFERENCE | The unique value indicating the fault logbook reference value. |
| DEFER_REF_SDESC | The deferral reference for the fault deferral. This field becomes mandatory for MEL type faults (REF_FAIL_SEV.SEV_TYPE_CD = 'MEL'). |
| OP_RESTRICTION_LDESC | A text value containing any operational restrictions relating to the fault. DEFER_CD_SDESC: The text value containing the deferral authorization value. |

| Column | Description |
|---|---|
| DEFER_CD_SDESC | The deferral authorization. This is an optional text field that applies only to MEL type faults. It can be used to specify an external document that authorizes the deferral. |
| FAULT_LOG_TYPE_CD | A reference term field to indicate the type of logbook entry. |
| FRM_CD | An optional free-form text field to store a troubleshooting FRM Code from the Fault Reporting Manual. |
| PN(x)_PART_NO_OEM | One of the part numbers in part requirement for a fault.<br>Parts can come from the current assembly, the common hardware assembly, and from related sub-assemblies (if permitted by the configuration parameter ALLOW_UNRELATED_PART_GROUP_IN_PART_REQUIREMENT). |
| PN(x)_MANUFACT_CD | The manufacturer of the listed part in part requirement for a fault. |
| PN(x)_IPC_REF_CD | The configuration slot onto which the part is installed. Only applicable for ASSY and TRK parts. It will be ignored for BATCH, SER, and KIT parts. |
| PN(x)_POS_CD | The position code on the configuration slot. Only applicable for ASSY and TRK parts. It will be ignored for BATCH, SER, and KIT parts. |
| PN(x)_INST_QT | The quantity of a part installed in part requirement. The default quantity is 1 when not provided. |
| PN(x)_RMVD_QT | The quantity of a part removed in part requirement. The default quantity is 1 when not provided. |
| RECORD_ID | The unique row identifier, which is sequence-generated. Leave this field blank. |
| PROCESSING_NOTE | An optional processing note. |

## Fault data processing tables

The data loaded in fault staging tables is used by the following processing tables:

- AL_PROC_OPEN_DEFERRED_FAULT
- AL_PROC_ODF_PART_REQUIREMENT

# Validating fault data

Once the staging tables have been populated, you must validate the data and check for any problems or errors. Populating and validating is typically an iterative process; you populate the staging tables, validate the data, resolve the errors by altering the data in the staging tables, and then validate again. You repeat the process until the validation succeeds.

You can validate fault data in two ways:

- Using .bat scripts
- Using packages

## Validating fault data using .bat file

**Steps**

1. Validate the data in the staging tables by running the following script file:
   `<installation_folder>\dataload\script_Actuals\odf\b.validate_odf.bat`

   Depending on the amount of data you have to import, this step might take several minutes to complete.

   When the script completes, it produces an error list: a file called `odf_errors.log`, located in the `dataload\script_Actuals\odf\logs` folder.

   The error list provides a summary of the records that were successfully validated and those in which errors were found.
2. To see the validation results, query the AL_PROC_RESULT table. This table can be joined to the AL_OPEN_DEFERRED_FAULT table using the record_id column. Error message descriptions are stored in the DL_REF_MESSAGE table.

   Sample query:

```
SELECT
   al_open_deferred_fault.record_id,
   al_proc_result.result_cd,
   dl_ref_message.tech_desc,
   al_open_deferred_fault.*
FROM
   al_open_deferred_fault
   INNER JOIN al_proc_result ON
      al_proc_result.record_id = al_open_deferred_fault.record_id
   LEFT OUTER JOIN dl_ref_message ON
      dl_ref_message.result_cd = al_proc_result.result_cd
UNION
SELECT
   al_open_deferred_fault.record_id,
   al_proc_result.result_cd,
   dl_ref_message.tech_desc,
   al_open_deferred_fault.*
```

```
FROM
   al_open_deferred_fault
   INNER JOIN al_proc_odf_part_requirement ON
      al_proc_odf_part_requirement.fault_record_id =
 al_open_deferred_fault.record_id
   INNER JOIN al_proc_result ON
      al_proc_result.record_id =
 al_proc_odf_part_requirement.record_id
   LEFT OUTER JOIN dl_ref_message ON
      dl_ref_message.result_cd = al_proc_result.result_cd;
```

## Validating fault data using packages

**Package name:**

AL_OPEN_DEFERRED_FAULT_PKG

**Procedure name:**

VALIDATE

**Input arguments:**

- aib_allornone: A Boolean value controlling partial load of data when validation errors are present.  False means partial commit mode is enabled. True (default) means all-or-none commit mode is enabled.
- ain_gather_stats_bool: A number value that turns gather stats on or off.  The default is 0 (do not gather stats)

**Output arguments:**

- aon_retcode:
  - 1: Data is valid and the package ran successfully
  - -1: Error occurred during execution
- aov_retmsg: Outcome description

## Error handling for fault data

If there are any errors encountered while validating the fault data, appropriate error codes will be added to the following table:

- al_proc_result

You can view the definitions for the error messages in the following table:

- dl_ref_message

# Importing fault data

Once the data has been validated, it can be imported into CAMO.

You can import fault data in two ways:

- Using .bat scripts
- Using packages

## Importing fault data using .bat file

**Steps**

Import the data from the staging tables by running the script file: `<installation_folder>\dataload\script_Actuals\odf\c.import_odf.bat`

## Importing fault data using packages

**Package name:**

AL_OPEN_DEFERRED_FAULT_PKG

**Procedure name:**

IMPORT

**Input arguments:**

- ain_gather_stats_bool: A number value that turns gather stats on or off.  The default is 0 (do not gather stats)

**Output arguments:**

- aon_retcode:
  - 1: Data is valid and the package ran successfully
  - -1: Error occurred during execution
- aov_retmsg: Outcome description

# 7

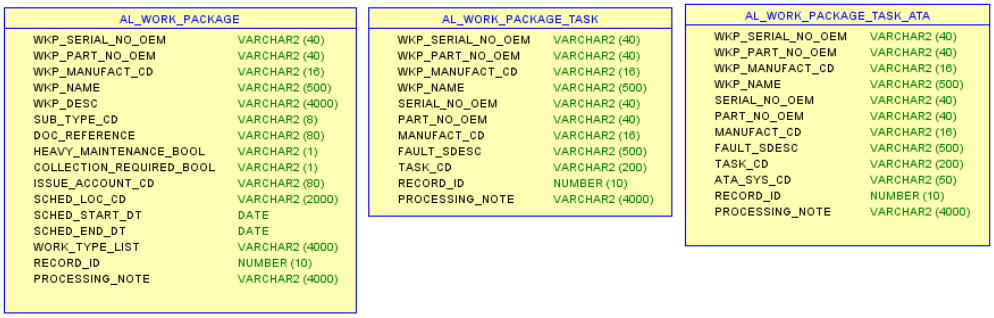
# Loading work packages

After loading the data for inventory, tasks and faults, you can load the data for work packages.

The Work Package staging area allows the bundling of tasks and faults into a work package applicable to the maintenance of aircraft and aircraft components. Setup of this data creates a work package and pushes it forward to an ACTV lifecycle state, but not to a workscope committed state. Aircraft work packages (CHECKs) are created at an internal site. External work packages (CHECKs) are currently not supported.

## Populate Work Package Staging Tables

In order to use the Actuals Loader, you must first populate the staging tables. These staging tables contain all of the data that will be used by Actuals Loader for validation and importing.



| AL_WORK_PACKAGE | |
|---|---|
| WKP_SERIAL_NO_OEM | VARCHAR2 (40) |
| WKP_PART_NO_OEM | VARCHAR2 (40) |
| WKP_MANUFACT_CD | VARCHAR2 (16) |
| WKP_NAME | VARCHAR2 (500) |
| WKP_DESC | VARCHAR2 (4000) |
| SUB_TYPE_CD | VARCHAR2 (8) |
| DOC_REFERENCE | VARCHAR2 (80) |
| HEAVY_MAINTENANCE_BOOL | VARCHAR2 (1) |
| COLLECTION_REQUIRED_BOOL | VARCHAR2 (1) |
| ISSUE_ACCOUNT_CD | VARCHAR2 (80) |
| SCHED_LOC_CD | VARCHAR2 (2000) |
| SCHED_START_DT | DATE |
| SCHED_END_DT | DATE |
| WORK_TYPE_LIST | VARCHAR2 (4000) |
| RECORD_ID | NUMBER (10) |
| PROCESSING_NOTE | VARCHAR2 (4000) |

| AL_WORK_PACKAGE_TASK | |
|---|---|
| WKP_SERIAL_NO_OEM | VARCHAR2 (40) |
| WKP_PART_NO_OEM | VARCHAR2 (40) |
| WKP_MANUFACT_CD | VARCHAR2 (16) |
| WKP_NAME | VARCHAR2 (500) |
| SERIAL_NO_OEM | VARCHAR2 (40) |
| PART_NO_OEM | VARCHAR2 (40) |
| MANUFACT_CD | VARCHAR2 (16) |
| FAULT_SDESC | VARCHAR2 (500) |
| TASK_CD | VARCHAR2 (200) |
| RECORD_ID | NUMBER (10) |
| PROCESSING_NOTE | VARCHAR2 (4000) |

| AL_WORK_PACKAGE_TASK_ATA | |
|---|---|
| WKP_SERIAL_NO_OEM | VARCHAR2 (40) |
| WKP_PART_NO_OEM | VARCHAR2 (40) |
| WKP_MANUFACT_CD | VARCHAR2 (16) |
| WKP_NAME | VARCHAR2 (500) |
| SERIAL_NO_OEM | VARCHAR2 (40) |
| PART_NO_OEM | VARCHAR2 (40) |
| MANUFACT_CD | VARCHAR2 (16) |
| FAULT_SDESC | VARCHAR2 (500) |
| TASK_CD | VARCHAR2 (200) |
| ATA_SYS_CD | VARCHAR2 (50) |
| RECORD_ID | NUMBER (10) |
| PROCESSING_NOTE | VARCHAR2 (4000) |

* Denotes columns that are mandatory.

**Denotes columns that are conditionally mandatory.

### AL_WORK_PACKAGE

This table lists all work packages that are loaded into the CAMO database.

| Column | Description |
|---|---|
| *WKP_SERIAL_NO_OEM | Serial number of the inventory assigned to the work package. |
| *WKP_PART_NO_OEM | Part number of the inventory assigned to the work package. |
| **WKP_MANUFACT_CD | Manufacturer of the inventory assigned to the work package. |
| *WKP_NAME | Full name of the work package. |
| WKP_DESC | A description of the work package. |
| SUB_TYPE_CD | Code value representing the sub type of the work package. Must be a valid entry in the REF_TASK_SUBCLASS table.<br>• For on-wing work packages this subclass must a parent class of CHECK.<br>• For off-wing work packages this subclass must have a parent class of RO. |
| DOC_REFERENCE | Document reference for the work package. |
| HEAVY_MAINTENANCE_BOOL | Boolean value indicating the work package is part of a "heavy maintenance check". |
| COLLECTION_REQUIRED_BOOL | Boolean value indicating that the task's job card has been physically collected. |
| ISSUE_ACCOUNT_CD | The issue to account of the work package. |
| SCHED_LOC_CD | The location where the work package is scheduled to be performed. |
| *SCHED_START_DT | The scheduled start date of the work package. |
| *SCHED_END_DT | The scheduled end date of the work package. |
| WORK_TYPE_LIST | List of comma-separated code values representing the work type of the work package. Each item in the list must be a valid entry in the REF_WORK_TYPE table. |
| RECORD_ID | Identifies row uniquely. Sequence-generated unique Id. Users should leave this field blank. |

| Column | Description |
|---|---|
| PROCESSING_NOTE | Optional processing note. |

## AL_WORK_PACKAGE_TASK

This table lists all of the tasks/faults collected as part of the work package. When the work package contains a fault, the fault_desc is populated. When the work package contains a baseline BLOCK or REQUIREMENT, task_cd is populated.

| Column | Description |
|---|---|
| *WKP_SERIAL_NO_OEM | Serial number of the inventory assigned to the work package. |
| *WKP_PART_NO_OEM | Part number of the inventory assigned to the work package. |
| **WKP_MANUFACT_CD | Manufacturer of the inventory assigned to the work package. |
| *WKP_NAME | Full name of the work package. |
| *SERIAL_NO_OEM | Serial number of the inventory collected as part of the work package. |
| *PART_NO_OEM | Part number of the inventory collected as part of the work package. |
| *MANUFACT_CD | Manufacturer of the inventory collected as part of the work package. |
| TASK_CD | The task collected as part of the work package. Either TASK_CD or FAULT_SDESC must be populated. |
| FAULT_SDESC | Fault description of the fault collected as part of the work package. Either TASK_CD or FAULT_SDESC must be populated. |
| RECORD_ID | Identifies row uniquely. Sequence-generated unique Id. Users should leave this field blank |
| PROCESSING_NOTE | Optional processing note |

## AL_WORK_PACKAGE_TASK_ATA

This table lists all of the tasks/faults assigned to a SYS configuration slot in the equipment hierarchy. When the work package contains a fault, the fault_desc is populated. When the work package contains a BLOCK or REQUIREMENT, task_cd is populated.

| Column | Description |
|---|---|
| *WKP_SERIAL_NO_OEM | Serial number of the inventory assigned to the work package. |
| *WKP_PART_NO_OEM | Part number of the inventory assigned to the work package. |
| **WKP_MANUFACT_CD | Manufacturer of the inventory assigned to the work package. |
| *WKP_NAME | Full name of the work package. |
| *SERIAL_NO_OEM | Serial number of the inventory collected as part of the work package. |
| *PART_NO_OEM | Part number of the inventory collected as part of the work package. |
| **MANUFACT_CD | Manufacturer of the inventory collected as part of the work package. |
| TASK_CD | The task collected as part of the work package. Either TASK_CD or FAULT_SDESC must be populated. |
| FAULT_SDESC | Fault description of the fault collected as part of the work package. Either TASK_CD or FAULT_SDESC must be populated. |
| *ATA_SYS_CD | The ATA system against which tasks or faults are defined. |
| RECORD_ID | Identifies row uniquely. Sequence-generated unique Id. Users should leave this field blank |
| PROCESSING_NOTE | Optional processing note |

# Work package data processing tables

The data loaded in work packages staging tables is used by the following processing tables:

- al_proc_work_package
- al_proc_work_package_task

# Validating work package data

Once the staging tables have been populated, you must validate the data and check for any problems or errors. Populating and validating is typically an iterative process; you populate the staging tables, validate the data, resolve the errors by altering the data in the staging tables, and then validate again. You repeat the process until the validation succeeds.

You can validate work package data in two ways:

- Using .bat scripts
- Using packages

## Validating work package data using .bat file

**Steps**

Validate the data in the staging tables by running the following script file:
`<installation_folder>\dataload\script_Actuals\workpackage\b.validate_work_package.bat`

Depending on the amount of data you have to import, this step might take several minutes to complete.

When the script completes, it produces an error list:

`dataload\script_Actuals\workpackage\logs\work_package_errors.log`.

The error list provides a summary of the records that were successfully validated and those in which errors were found.

## Validating work package data using packages

**Package name:**

AL_WORK_PACKAGE_PKG

**Procedure name:**

VALIDATE

**Input arguments:**

- aib_allornone: Indicates if "All-or-None" or "Partial" mode.

- ain_gather_stats_bool: Turn gather stats on or off.

**Output arguments:**

- aon_retcode:
  - 1: This error message means that validate procedure executed successfully without any unexpected errors. It doesn't mean that data is valid or not.
  - -1: Error occurred during execution.
- aov_retmsg: Outcome description.

## Error handling for work packages

If there are any errors encountered while validating the usage data, appropriate error codes will be added to the following table:

- al_proc_result

You can view the definitions for the error messages in the following table:

- dl_ref_message

# Importing work package data

Once the data has been validated, it can be imported into CAMO.

You can import work package data in two ways:

- Using .bat scripts
- Using packages

## Importing work package data using .bat file

**Steps**

Run the following script file:

```
<installation_folder>\dataload\script_Actuals\workpackage
\c.import_work_package.bat.
```

## Importing work package data using packages

**Package name:**

AL_WORK_PACKAGE_PKG

**Procedure name:**

IMPORT

**Input arguments:**

- ain_gather_stats_bool: Turn gather stats on or off.

**Output arguments:**

- aon_retcode:
  - 1: This error message means that the validate procedure executed successfully without any unexpected errors. It doesn't indicate whether or not thedata is valid.
  - -1: Error occurred during execution.
- aov_retmsg: Outcome description.

# 8

# Loading purchase orders

The Actuals Loader can be used to load many forms of purchase orders into CAMO. There are five types of purchase orders that are handled by these staging areas:

- Purchase Orders (Part Purchase)
- Repair Orders
- Exchange Orders
- Borrow Orders
- Consignment Orders

**Note** Only orders in open issued stage are in scope to be loaded by Actuals Loader.

## Loading purchase orders .csv files

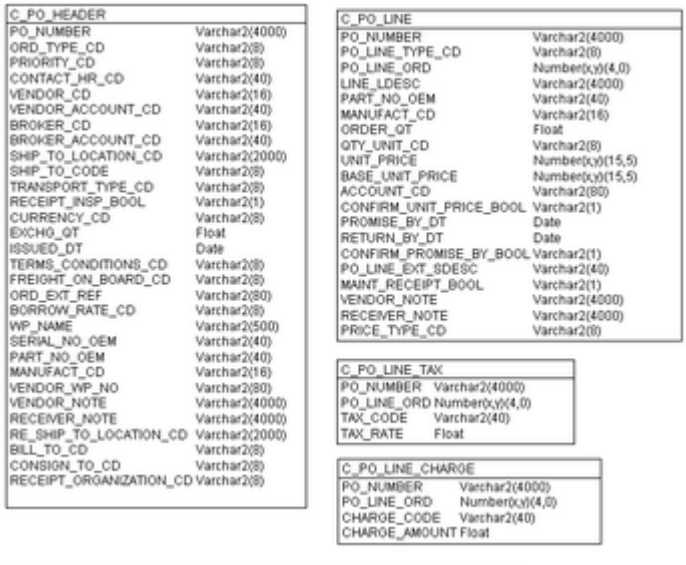
**Steps**

Fill out your purchase order *.csv files following these example templates found in `<installation_folder>\dataload\script_Actuals\po\data`:

- `C_PO_HEADER_sample.csv`
- `C_PO_LINE_CHARGE_sample.csv`
- `C_PO_LINE_sample.csv`
- `C_PO_LINE_TAX_sample.csv`

# Populating purchase order staging tables

In order to use the Actuals Loader, you must first populate the staging tables. These staging tables contain all of the data that will be used by Actuals Loader for validation and importing.




* Denotes columns that are mandatory.

**Denotes columns that are conditionally mandatory.

## C_PO_HEADER

This table lists the header information for the purchase order. There is one row for each purchase order.

*C_PO_HEADER columns*

| Column | Description |
|---|---|
| *PO_NUMBER | The purchase order number. |
| *ORD_TYPE_CD | The order type. Must be a valid entry in the REF_PO_TYPE table. |
| *PRIORITY_CD | Priority code for the purchase order. Must be a valid entry in the REF_REQ_PRIORITY table. |
| *CONTACT_HR_CD | The HR that is the purchasing contact for this purchase order. |

| Column | Description |
|---|---|
| *VENDOR_CD | The vendor for the purchase order. |
| VENDOR_ACCOUNT_CD | The vendor account that is being used for the purchase order. |
| BROKER_CD | The broker for the purchase order. This is for repair orders only. |
| BROKER_ACCOUNT_CD | The broker account that is being used for the purchase order. This is for repair orders only. |
| *SHIP_TO_LOCATION_CD | The location where the repair take place. |
| *SHIP_TO_CODE | The Ship to Code of a purchase order. |
| TRANSPORT_TYPE_CD | The type of transportation the vendor is to use when they ship the parts. Must be a valid entry in the REF_TRANSPORT_TYPE table. |
| RECEIPT_INSP_BOOL | The flag indicates if everything on this order needs to go through inspection upon receipt. |
| *CURRENCY_CD | The currency for the purchase order. Must be a valid entry in the REF_CURRENCY table. |
| *EXCHG_QT | The exchange rate at the time of the PO issue. |
| *ISSUED_DT | The date that the purchase order was issued. |
| TERMS_CONDITIONS_CD | The payment terms and conditions for the purchase order. Must be a valid entry in the REF_TERMS_CONDITIONS table. |
| FREIGHT_ON_BOARD_CD | Used to indicate whether the purchaser takes ownership before or after the shipment has started. Must be a valid entry in the REF_FOB table. |
| ORD_EXT_REF | The PO reference number. |
| BORROW_RATE_CD | The borrow rate code. Must be a valid entry in the REF_BORROW_RATE table. This is for borrow orders only. |
| WP_NAME | The external work package's name. This is for repair orders only. |

| Column | Description |
|---|---|
| SERIAL_NO_OEM | The serial number of inventory. This is for repair orders only. |
| PART_NO_OEM | The part number. This is for repair orders only. |
| MANUFACT_CD | The part's manufacturer code. This is for repair orders only. |
| VENDOR_WP_NO | The correspondent for the work order of an external repair shop. |
| VENDOR_NOTE | The note that appears on the printed purchase order that is sent to the vendor. |
| RECEIVER_NOTE | The notes to the receiver regarding this purchase order. |
| RE_SHIP_TO_LOCATION_CD | The location to where the shipment is re-expedited. |
| BILL_TO_CD | The person/group to whom to bill the order. |
| CONSIGN_TO_CD | The person/group to who to consign the order. |
| RECEIPT_ORGANIZATION_CD | Identifies the originating organization of the order. |

## C_PO_LINE

This table is used to store the information for each line on a purchase order. There are one or more rows for each C_PO_HEADER row.

*C_PO_LINE*

| Column | Description |
|---|---|
| *PO_NUMBER | The purchase order number. |
| PO_LINE_TYPE_CD | The line type. Must be a valid entry in the REF_PO_TYPE table. |
| *PO_LINE_ORD | The line order number. |
| LINE_LDESC | The description of the purchase order line. |

| Column | Description |
|---|---|
| PART_NO_OEM | The part number. |
| MANUFACT_CD | The part's manufacturer code. |
| *ORDER_QT | The quantity that is being ordered. |
| *QTY_UNIT_CD | The units of measure for the quantity that is being ordered. Must be a valid entry in the REF_QTY_UNIT table. |
| *UNIT_PRICE | The price of one unit of the purchased part. |
| BASE_UNIT_PRICE | The base unit price for the part. This is required for borrow orders only. |
| *ACCOUNT_CD | The financial account that is billed for the purchase item or services. This is used during the financial transaction that is recorded when the invoice is closed. |
| CONFIRM_UNIT_PRICE_BOOL | The boolean to indicate that the price for the unit is confirmed. |
| *PROMISE_BY_DT | The date by which the vendor promises the ordered part. |
| RETURN_BY_DT | The date that the part is expected to be returned. (Required for Exchange and Borrow Orders only) |
| CONFIRM_PROMISE_BY_BOOL | The boolean to indicate that the promised price for the unit has been confirmed. |
| PO_LINE_EXT_SDESC | The reference to PO Line number in an external system. |
| MAINT_RECEIPT_BOOL | The flag that indicates whether the part should be picked up by maintenance. |
| VENDOR_NOTE | The note to the vendor that is attached to this particular line, rather than the general note for the purchase order. |
| RECEIVER_NOTE | The note that is sent to the receiver once the purchased items arrive. |

## C_PO_LINE_TAX

This table is used to store the information for each tax on a PO line. There are zero or more rows for each C_PO_LINE row.

*C_PO_LINE_TAX*

| Column | Description |
|---|---|
| *PO_NUMBER | The purchase order number. |
| *PO_LINE_ORD | The line order number. |
| *TAX_CODE | The unique code for tax. |
| *TAX_RATE | The tax rate. |

## C_PO_LINE_CHARGE

This table is used to store the information for each charge on a purchase order line. There are zero or more rows for each C_PO_LINE row.

*C_PO_LINE_CHARGE*

| Column | Description |
|---|---|
| *PO_NUMBER | The purchase order number. |
| *PO_LINE_ORD | The line order number. |
| *CHARGE_CODE | The unique code for charge. Must be a valid entry in the CHARGE table. |
| *CHARGE_AMOUNT | The charge amount applied to that purchase order line. |

# Purchase order data processing tables

The data loaded in inventory staging tables is used by the following processing tables:

- c_proc_po_header
- c_proc_po_line
- c_proc_po_line_charge
- c_proc_po_line_tax

# Validating purchase order data

Once the staging tables have been populated, you must validate the data and check for any problems or errors.

Populating and validating is typically an iterative process; you populate the staging tables, validate the data, resolve the errors by altering the data in the staging tables, and then validate again. You repeat the process until the validation succeeds.

You can validate purchase order data in two ways:

- Using .bat scripts
- Using packages

## Validating purchase order data using .bat file

**About this task**

All errors must be resolved and corrected in the staging table before you can import the data. If there are any warnings, they will show up in italics and do not need to be corrected before performing the import.

**Steps**

1. Validate the data in the staging tables by running the following script file:

   ```
   <installation_folder>\dataload\script_Actuals\po
   \b.validate_purchase_order.bat
   ```

   Depending on the amount of data you have to import, this step might take several minutes to complete.

   When the script completes, a browser window opens showing the results of the validation
2. If a popup window appears at the bottom of your browser window, click **Allow blocked content**.
3. To see more information about which records contain a particular error, click the Error Code. There is also a listing of all of the validation errors found in each staging table as follows.

The report summarizes the data from each staging table as follows:

- Staging: indicates the staging table name
- Processed: indicates how many records were loaded
- Valid: indicates how many loaded records were valid
- Error: indicates how many loaded records were invalid due to errors
- Percentage: indicates the percentage of valid records out of the total number of records

## Validating purchase order data using packages

**Package name:**

C_PO_IMPORT

**Procedure name:**

VALIDATE_PURCHASE_ORDER

**Input arguments:**

None

**Output arguments:**

- aon_retcode:
  - 1: It successfully executed this process. It does not indicate whether or not your data is valid.
  - -1: It means the package failed during execution.
- aov_retmsg: Outcome description

## Error handling for purchase order data

If there are any errors encountered while validating the purchase order data, appropriate error codes will be added to the following table:

- c_proc_po_header
- c_proc_po_line
- c_proc_po_line_change
- c_proc_po_line_tax

You can view the definitions for the error messages in the following table:

- dl_ref_message

# Importing purchase order data

Once the data has been validated, it can be imported into CAMO.

## Importing purchase order data using .bat file

**Steps**

Run the following script file:

```
<installation_folder>\dataload\script_Actuals\po
\c.import_purchase_order.bat
```

When the script completes, a browser opens an Actuals Loader Report displaying the results of the import.

The report summarizes the data from each staging table as follows:

- Staging: indicates the staging table name
- Processed: indicates how many records were loaded
- Imported: indicates how many loaded records were imported
- Error: indicates how many loaded records were not able to be imported due to errors
- Percentage: indicates the percentage of imported records out of the total number of records

## Importing purchase order data using packages

**Package name:**

C_PO_IMPORT

**Procedure name:**

IMPORT_PURCHASE_ORDER

**Input arguments:**

None

**Output arguments:**

- aon_retcode:
  - 1: It successfully execute this process. It does not indicate whether or not your data is valid.
  - -1: Means the package failed during execution.
- aov_retmsg: Outcome description

# 9

# Loading inventory original equipment

After loading the data for inventory, you can associate it with its original equipment.

## Loading inventory original equipment .csv files

**About this task**

In order to use the Actuals Loader, you must populate the staging tables described in the next section. Instead of using SQL to fill in these staging tables, Actuals Loader provides the option to fill in .csv files and then load those .csv files into the staging tables.

**Steps**

1. Fill out your AL_IOE_ASSMBL.csv files following these example templates found in `<installation_folder>\dataload\script_Actuals\warranty\data`
   - `AL_IOE_ASSMBL_sample.csv`
2. Copy your completed `AL_IOE_ASSMBL.csv` file to the same directory (`<installation_folder>\dataload\script_Actuals\warranty\data`).

   Sample completed file:

| CSV_FILENAME | STG_TABLE_NAME | NUMBER_TITLE_ROWS | DATE_FORMAT | EXEC_BOOL |
|---|---|---|---|---|
| AL_OEM_ASSMBL_sample.csv | AL_OEM_ASSMBL | 1 | MM/DD/YYYY HH24:MI | 1 |

   - CSV_FILENAME (column A): List AL_IOE_ASSMBL.csv file that you would like to load into the staging tables. Make sure that none of these are blank, otherwise the loading script will fail
   - STG_TABLE_NAME (column B): The corresponding Oracle staging table name. Do not change these values
   - NUMBER_TITLE_ROWS (column C): List the number of rows used for titles in your file from column A. If the data in your AL_IOE_ASSMBL.csv data file does not start at the 2nd row,

increment this number to show how many title rows to skip before the inventory loading script looks for your data

- DATE_FORMAT (column D): List the format of the dates used in your file from column A. This doesn't apply for this module
- EXEC_BOOL (column E): This is a binary value. If the value is a 1, the data loading script will load the file's data into the staging tables. If the value is a 0, the data loading script will skip over this file and will not load it

3. Load the inventory files into the staging tables by running the script file `<installation_folder>\dataload\script_Actuals\usage\a.load_warranty.bat`

When the script completes, it displays a command window showing a status of the loading process. Usually it says 'BUILD SUCCESSFUL' for successful import and 'BUILD FAILED' for if it load failed displayed at the end of the command window. Check staging table AL_IOE_ASSMBL to see if data has been loaded.

## Populating inventory original equipment staging table

In order to use the Actuals Loader, you must first populate the staging tables. These staging tables contain all of the data that will be used by Actuals Loader for validation and importing.

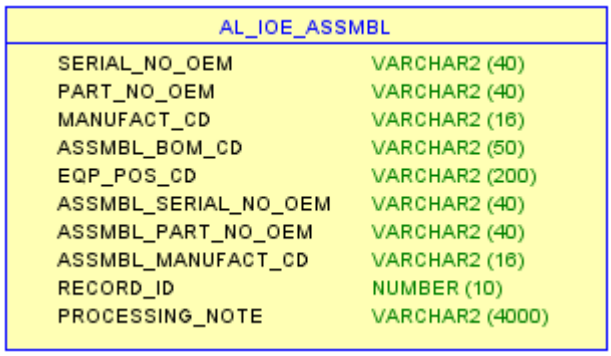
* Denotes columns that are mandatory.

**Denotes columns that are conditionally mandatory.

### AL_IOE_ASSMBL

The AL_IOE_ASSMBL staging table allows components to be associated with their original assemblies or aircraft. The components can be provided in the staging table in two different ways:

1. When the serial number and part number of the component are known, they can be specified in the serial_no_oem, part_no_oem and manufact_cd columns. This allows loose or installed components to be associated with their original assemblies.
2. If the serial number is not known and the component is still installed on its original assembly, then the component can be looked up automatically by specifying the configuration slot and position in the assmbl_bom_cd and eqp_pos_cd columns.




| Column | Description |
|---|---|
| *SERIAL_NO_OEM | The Component's Serial Number |
| * PART_NO_OEM | Identifies the Part Number of the component |
| MANUFACT_CD | Identifies the Manufacturer of the component |
| **ASSMBL_BOM_CD | Identifies config slot for the component. This column and the eqp_pos_cd column can be used instead of the serial_no_oem, part_no_oem and manufact_cd when the serial number is not known. |
| **EQP_POS_CD | The position code where this component is installed. This column and the assmbl_bom_cd column can be used instead of the serial_no_oem, part_no_oem and manufact_cd when the serial number is not known. |
| *ASSMBL_SERIAL_NO_OEM | The original equipment manufacturer's serial number |
| * ASSMBL_PART_NO_OEM | The Part Number of the OEM assembly |
| ASSMBL_MANUFACT_CD | The Manufacturer Code of the OEM assembly |

## Inventory original equipment data processing tables

The data loaded in inventory original equipment staging tables is used by the following processing tables:

- al_proc_ioe_assmbl

# Validating inventory original equipment data

Once the staging tables have been populated, you must validate the data and check for any problems or errors.

Populating and validating is typically an iterative process; you populate the staging tables, validate the data, resolve the errors by altering the data in the staging tables, and then validate again. You repeat the process until the validation succeeds.

You can validate inventory original equipment data in two ways:

- Using .bat scripts
- Using packages

## Validating inventory original equipment using .bat file

**About this task**

Once the staging tables have been populated, you must validate the data and check for any problems or errors. Populating and validating is typically an iterative process; you populate the staging tables, validate the data, resolve the errors by altering the data in the staging tables, and then validate again. You repeat the process until the validation succeeds.

**Steps**

1. Run the following script file.

   ```
   <installation_folder>\dataload\script_Actuals\warranty
   \b.validate_warranty.bat
   ```

   Depending on the amount of data you have to import, this step might take several minutes to complete.

   When the script completes, it will say "BUILD SUCCESSFUL". This just means that validation has been successfully completed. In case the process failed, it will display error which can further diagnosed upon careful analysis.

2. To see the validation results, query the AL_PROC_IOE_ASSMBL and the AL_PROC_RESULT table, Inner Join the above tables by RECORD_ID and select all key values from AL_PROC_IOE_ASSMBL.

   Sample query:

   ```
   SELECT result_cd, al_proc_ioe_assmbl.* FROM al_proc_result
   INNER JOIN al_proc_ioe_assmbl
      ON al_proc_ioe_assmbl.record_id = al_proc_result.record_id
   ```

## Validating inventory original equipment data using packages

**Package name:**

AL_IOE_ASSMBL_IMPORT

**Procedure name:**

VALIDATE

**Input arguments:**

- aib_allornone: It is a boolean value allowing partial load whether there are data issues or not.

**Output arguments:**

- aon_retcode:
  - 3: There were data validation errors during partial load.
  - 2: No data in staging table.
  - 1: Data processed successfully.
  - -1: Uncaught Exception error occurred.
  - -2: All or none; at least one record failed validation.
- aov_retmsg: Outcome description

## Error handling for inventory original equipment data

If there are any errors encountered while validating the inventory original equipment data, appropriate error codes will be added to the following table:

- al_proc_ioe_assmbl

You can view the definitions for the error messages in the following table:

- dl_ref_message

# Importing inventory original equipment data

Once the data has been validated, it can be imported into CAMO.

## Importing inventory original equipment data using .bat file

**Steps**

1. Run the following script file:

   ```
   <installation_folder>\dataload\script_Actuals\warranty
   \c.import_warranty.bat
   ```

When the script completes, it displays a command window showing a status of the loading process. The command window will display ‘BUILD SUCCESSFUL’ if the import was successful, or ‘BUILD FAILED’ if the import failed.

2. Check the INV_INV_OEM_ASSMBL table to verify that the data has been loaded correctly.

## Importing inventory original equipment data using packages

**Package name:**

AL_IOE_ASSMBL_IMPORT

**Procedure name:**

IMPORT

**Input arguments:**

- aib_allornone: It is a boolean value allowing partial load whether or not there aredata issues.

**Output arguments:**

- aon_retcode:
  - 3: There were data validation errors during partial load.
  - 2: No data in staging table.
  - 1: Data processed successfully.
  - -1: Uncaught Exception error occurred.
  - -2: All or none, at least one record failed validation.
- aov_retmsg: Outcome description

## 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.