IFS Aviation Maintenance

Data Loading Installation Guide

ICAM 26.2



Document Revision: 26.2-1

Publication Date: November 03, 2026

Contents

1 Overview of data loading 3
Loading your baseline data 3
Loading your actuals data 5

2 Prerequisites 6

3 Configuring runtime.properties 7
Configuring Actuals Loader for Parallel Execution 9
Optimizing the parallelism.degree property 9
Optimizing the parallelism.chunk.size property 10

4 Installing Data Loading Tool 11

5 Uninstalling Data Loading Tool 12

6 Baseline Loader page 13

7 Switching between multiple CAMO user schemas 14

1

Overview of data loading

This guide is intended for Business Architects and Analysts who are responsible or involved in the migration of data from legacy systems to the CAMO database.

Data loading consists of migrating the data from your organization's legacy systems to the database, so that personnel in various departments can use CAMO to plan and track the maintenance of your aviation assets.

To complete data loading, you must load your baseline data first, and then load your actuals data:

- **Baseline data** refers to all the configuration data and reusable template data for aircraft, other types of assets, maintenance tasks, and materials management that are required for CAMO and all integrations of CAMO with external systems to work properly.
- **Actuals data** is the information that relates to a specific inventory item; its serial number, manufactured date, and accumulated usage; the maintenance tasks that have been completed on the item and the tasks that are upcoming; the deferred faults that exist on the inventory; the purchase orders for inventory items; and so on.

Data loading tools are included to assist in both of these activities. The data loading tools are the Baseline Loader for loading baseline data, and the Actuals Loader for loading actuals data. Both tools use similar approaches of loading data from temporary tables into the CAMO database. These temporary tables are called staging tables or collection tables Actuals Loader.

When you install Baseline Loader or Actuals Loader, the staging (collection) tables are automatically created in the database, as well as the processing tables that both tools use during the validation and import processes.

Loading your baseline data

The CAMO Baseline Loader provides analysts with data format specifications for the baseline, and software tools to support the roll-out of your organization's baseline data.

You load baseline data using the Baseline Loader Web interface. An alternative method is also provided, which lets you import some types of baseline data from comma-separated-values (CSV) files by using scripts.

Because of inherent dependencies between different pieces of data, you must load the different types of data that make up your baseline in a specific order. The types of data to load are called subject areas. The subject areas are organized in lettered groups, groups A through F. You must load the data for each group in the order specified in this guide. Also, within each group, you must load the data for the different subject areas in a prescribed sequence. For example in group A, you must import the list of manufacturers (one subject area) before you can import data about a part (another subject area of the same group) that contains the name of the part manufacturer.

Baseline Loader includes additional scripts you can use to create replacement task definitions for assets that require such tasks (sub-assemblies and tracked parts) and to add new material storage locations to existing airports. Also offered is functionality to extract and export baseline data from the database.

The Baseline Loader data loading tools are provided for the initial loading of data required to set up the baseline. You can also use the Baseline Loader data loading tools after the initial data load, for example, to update baseline data related to procurement agreements and stock, and to add store and bin locations.

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

1. Verify that the software requirements and prerequisite conditions are met.
2. Install the Data Loader tools in your database.
3. Populate the Baseline Loader staging tables with the baseline data.

   Based on the CAMO solution you are implementing, you must add the relevant baseline data to the staging tables. For some types of data records, you can put the required data into CSV files instead of populating staging tables.
4. Access the Baseline Loader interface.
5. Validate the data in the staging tables before loading the data.

   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.
6. Import the successfully validated data.

   You import the data for the different subject areas and functional areas in a specific sequential order. For data in staging tables, you use the Baseline Loader interface to import the data. For data that you can load from CSV files, you use loading scripts provided with Baseline Loader.

Loading your actuals data

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

Before loading aircraft data, referred to as actuals data, into CAMO, you must load your baseline data.

Actuals data refers to a specific aircraft tail number; its serial number, manufactured date, and accumulated usage; the list of sub-components on the aircraft; the maintenance tasks that have been completed and/or the tasks that are upcoming; the deferred faults; and so on.

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

1. Verify that the software requirements and prerequisite conditions are met.
2. Install the Data Loader tools, if not already installed in your CAMO database.
3. Populate the collection tables with the actuals data. Based on the CAMO solution you are implementing, you must add the relevant data to the staging tables.
4. Validate the data in the collection tables before loading the data. 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.
5. Import the successfully validated data into CAMO, in the prescribed order.

2

Prerequisites

The CAMO operational environment requires third-party software, such as the application server, database server, and more. To use Data Loading tool, and the other baseline loading scripts that are included with Data loading tool, you must have access to an Oracle client and sqlplus executable to run the database data and schema update scripts. For detailed information about the software requirements, including the specific release numbers of the third-party software and the supported operating systems, see the *CAMO Installation and Configuration Guide*.

The following conditions must be fulfilled before you install the data loading tool:

- Oracle is installed on the database server.
- The CAMO application server component is installed. See the *CAMO Installation and Configuration Guide*.
- The CAMO base database and level-10 solution data are installed. See the *CAMO Database Installation Guide*.
- Other releases of data loading tool are uninstalled using `remove-all.bat` in the `deinstall` folder.
- The *runtime.properties* file is configured properly.

3

Configuring runtime.properties

The properties contained in the *runtime.properties* file are required to install the staging, collection, and processing tables to the database, and to validate and load different subject areas (like inventory, tasks, etc.).

About this task

Each operation (installing tables, loading inventory, loading tasks, loading faults, loading work packages and purchase orders) uses a different subset of properties in the file.

For example, when you install the data loader tables in the database, only the properties that identify the Oracle connection information and the database information are used. When you load inventory data, additional properties that specify configuration options are used.

You can set the correct values for all properties at once, and have the file ready for all operations, or set or verify only the values of the properties you need for the operation you are about to perform.

Steps

1. Go to the following folder:

   `<database_install>\dataload`

   where `<database_install>` is the folder where you installed the CAMO database file set (default is `C:\mxi\database-<release_number>`).
2. In a text editor, open the `runtime.properties` file.
3. Set the correct values for the following properties. Note values are case insensitive and are enclosed in "<>" for readability:
   - **oracle.connect.string**: <Database alias as stated in tnsnames.ora, or the EZCONNECT string.>
   - **maintenix.username**: <Name of the CAMOv user schema.>
   - **maintenix.password**: <Password for the CAMO user schema.>

- **load.db.id**: Database identifier (DBID). Use this to override the default local database ID, and load the data with a different ID. Ensure that a matching entry exists in the MIM_DB table when setting this property. When no value is assigned to this property, the data is loaded using the MIM_LOCAL_DB database ID.
- **complete.assy.bool**: To load incomplete assemblies or use the CAMO Ready for Build feature, set to <NOT>. Otherwise set to <FULL>.
- **parallelism.degree**: See Section below, '***Configuring Actuals Loader for Parallel Execution***'.
- **parallelism.chunk.size**: See Section below, '***Configuring Actuals Loader for Parallel Execution***'.
- **dldr.module.name**: Used for setting the module name for versioning info in the database as well as for output messages when installing and deinstalling the Dataloading Tool. Also used by gradle to set the module name when creating a dataloader build file.
- **install.failonerror**: Controls if the batch files stop on failure when installing and deinstalling the Dataloader Tool. Also controls stop on failure when gathering schema stats for Actuals Loader modules ODF, PO, Usage, Warranty and WorkPackage. Controls stop on failure for Actuals Loader Task import and running the Missing Task report. Allowable values are <true> or <false>.
- **pause.batch.file**: For pausing the Actuals Loader batch files after running the load, validate or import. Will also control the pause for finalizing the inventory load. Values are <true> or <false>.
- **generate.report**: Determines if an HTML report of results is generated when Actual Loader batch files are run. Values are <true> or <false>.
- **gather.stats.bool:** Controls if the Actuals Loader and Baseline Loader will gather statistics on core CAMO tables during the validation and import process. The allowable values for this property are are <0> and <1>.
  - If the database already has significant actuals data (ex: loading second fleet), then the gather.stats.bool may be set to 0 if the existing database statistics will support the desired data loading performance
  - If the database does not have significant actuals data, then the gather.stat.bool must be set to 1 to avoid poor data loading performance.

> **Note** Table statistics will always be gathered for the data loading staging and processing tables regardless of setting of `gather.stat.bool`. This parameter only controls gathering statistics on core CAMO tables, however it requires that statistics must already exist for the database. Please ensure that statistics have been gathered manually after importing a database dump.

- **all.or.none.mode**: Controls how Actuals Loader data is imported following validation. If set to <True> all records must be successfully validated or none will be imported. If set to <False>, can have a partial load of data where successfully validated records are imported and unsuccessful records ignored. This applies to Actual Loader Warranty and Workpackage areas. Values are <true> or <false>.

Configuring Actuals Loader for Parallel Execution

To improve performance, Actuals Loader supports parallel processing in the following areas:

- Inventory
- Tasks
- Historical Usage

Parallelism is disabled by default in Actuals Loader, and it is recommended to leave it disabled unless performance is a specific concern and an Oracle DBA is available to optimize the parallelism settings. Parallelism must also be disabled when using RAC (Oracle's Real Application Clusters).

Parallelism is configured through the following properties in the runtime.properties file:

- **parallelism.degree:** This is the available threads for parallel processing. Set this to 1 to disable parallelism.
- **parallelism.chunk.size:** The amount of data that will be processes by each thread.

You can find the optimal parallelism properties for your data on your servers through testing using a realistic data set on a test server that has a similar hardware and configuration to the one that will be used during the final data loading.

Note DBAs should be involved to capture Oracle AWR reports or other server diagnostic reports during Data Loading runs. This information should be analyzed to identify issues with Oracle server resources and configuration.

Optimizing the parallelism.degree property

The computing power to run parallel processes is physically limited by the number of cores on that server. You can find the maximum number of cores using the following query:

SELECT cpu_core_count_current FROM v$license;

This is a guideline for the upper limit of parallelism. Modern Oracle servers could have 12 or more cores, however this level of parallelism is rarely achievable because other sub-systems (e.g. disk operations) would become a bottleneck. The maximum degree of parallelism is also limited by Oracle server’s configuration parameter "job_queue_processes".

SELECT value FROM v$parameter WHERE name = 'job_queue_processes';

Before considering a high level of parallelism, it is recommended to confirm (with the help of AWR reports and the assistance of a DBA) that your Oracle server is actually prepared for Bulk Loading operations. Oracle servers are often optimized for production operations that are not well suited for bulk data loading. A typical example is the number and size of redo log files, and the number of database writer processes. The default settings and typical recommendations are often related to production operation of the Oracle server, and may need to be increased by DBA if the AWR report reveals that log file switching stands out as a wait event during bulk loading. We recommend that DBAs should closely monitor the test runs and address shortages of resources or adjust the server configuration until the physical limitations of the hardware are reached.

Optimizing the parallelism.chunk.size property

Choosing the chunk size means finding the right balance. Each chunk has overhead to create and manage that chunk. If the chunk size is too small, it increase the total number of chunks and the overhead penalty can reduce the performance. On the other hand, if the chunk size is too large, then more memory is used and the large commit size can be hard on the undo and redo log files.

When creating chunks, Data Loader aligns the chunks by the highest inventory. It is useful to find out how many records in your dataset belong to the highest inventory. For example, if the largest aircraft being loaded has 5000 inventory items, then the chunk size should be at least this large. The optimal chunk size will be different when loading inventory, tasks and historic usage.

Your DBA can provide suggestions about a maximum commit size on your Oracle server to avoid exceeding the available resources. General guidance is to avoid having a chunk size that would cause uncommitted changes span across multiple redo log files.

Experience has shown that there is generally a wide range of acceptable values for the chunk size. As long as the chunk size is not too small, and not too large, then the "sweet spot" is quite wide and there isn’t much benefit in fine tuning it.

4

Installing Data Loading Tool

About this task

When you install Data loading tool, the staging, processing and collection tables required for loading baseline and actual data are automatically created in your CAMO database. These tables exist outside of the CAMO data model, and they are used to validate and process the data before importing the data into the CAMO tables.

To install the tables to the CAMO database and to use the Actuals Loader scripts to validate and import data, you must install the Actuals Loader files. The Actuals Loader files are included with the CAMO software delivery; they are part of the database file set.

Steps

1. Stop the CAMO application server.
2. Go to the following folder:

   `<database_install>\dataload\install`
3. Run install-all.bat (or install-all.sh if you are using a computer that runs on Linux or AIX).

   The script runs in a command prompt window. When you see the BUILD SUCCESSFUL message, the data loader database objects (tables, schema, data, pl/sql) are installed. If you see a BUILD FAILED message, check the logs folder and find `install-schema.txt` to further diagnose the error.
4. If necessary, inspect the installation log files for failure notifications. The log files are in `<database_install>\dataload\install\logs`.
5. To improve performance, ensure that Oracle Optimizer statistics have been generated on the database schema using the DBMS_STATS package.

5 Uninstalling Data Loading Tool

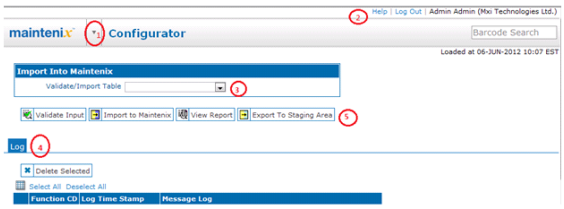
Steps

1. Go to the following folder:
   `<database_install>\dataload\deinstall`
2. Run `remove-all.bat` (or `deinstall-all.sh` if your server runs on Linux or AIX).
3. Inspect the uninstallation log file for failure notifications. The log file is in `<database_install> \dataload\deinstall\log`.

6

Baseline Loader page

Here is how the **Configurator** page looks like:



1. The CAMO menu drop-down arrow opens the list of menus that you can access.
2. In the upper-right hand corner are the **Help** and **Log Out** buttons, and your user name and role.
3. The **Validate/Import Table** drop-down list is where you select the type of data to validate, import, or export from the CAMO database.
4. The **Log** tab shows messages about the actions that were completed.
5. **Buttons** initiate actions.

7

Switching between multiple CAMO user schemas

For non-production environments, such as test environments, it is common to have a single Oracle database instance that stores multiple CAMO user schemas. Where multiple schemas exist, it is possible to switch between them in order to load actuals data for production and non-production schemas.

Steps

1. Obtain the required schema and workflow parameter values from your database administrator.
2. Modify the appropriate parameters in the runtime.properties file as outlined in the section 3, 'Configuring runtime.properties'.

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.