ALMAQSO Documentation and Reference Manual

Akimasa Nishida, Yuki Yoshimura

Dec 30, 2025

Contents:

1 Install

1.1 Pre-requisites

Before installing this library *almaqso*, please ensure that you have the following software installed on your system:

- Python (verified with version 3.13, however, you may use other Python 3.x versions)
- CASA (version 6.6.1 or later is recommended) with ALMA pipeline.
- analysisUtilites (latest version)

1.2 Installation

You can install the *almaqso* library using *pip*:

```
pip install almaqso
```

This will install the latest version of *almaqso* along with its dependencies.

2 Basic usage

Sample code to

- download J2000-1748 data in Band 4 of all cycles from ALMA archive,
- calibrate the data, and
- create MFS and Cube images using tclean of CASA

is shown below:

```
import sys
from almaqso import Almaqso

if __name__ == "__main__":
    almaqso = Almaqso(
        target="J2000-1748",
        band=4,
        cycle="",
        work_dir="your_work_dir/",
        casapath="/usr/local/casa/casa-6.6.6-17-pipeline-2025.1.0.35-py3.10.el8/bin/casa"
    )
    almaqso.process(
        n_parallel=2,
        do_tclean=True,
        tclean_mode=["mfs", "cube"],
    )
```

almaqso.Almaqso.process() has other useful options to control the processing and data storing.

3 Architecture

This chapter is basically for developers and new contributors of this library who want to understand the internal architecture and design of the library.

However, *System context* may give researchers an overview of this library.

3.1 System context

This section provides a general overview of the ALMAQSO system within user (researcher) and external system interactions.

Purpose

This diagram explains how ALMAQSO fits into the surrounding ecosystem and how researchers interact with external systems.

Audience

- Researchers using ALMAQSO
- New contributors
- Developers unfamiliar with CASA

Components

- **User**: Users who conduct data analysis using ALMAQSO. They are typically researchers in astronomy.
- **ALMAQSO**: This software system that provides automated data retrieval and analysis for ALMA calibration sources.
- **External Systems**: CASA, analysisUtils, ALMA Science Archive and local file systems that interact with ALMAQSO.

C4-Context Diagram

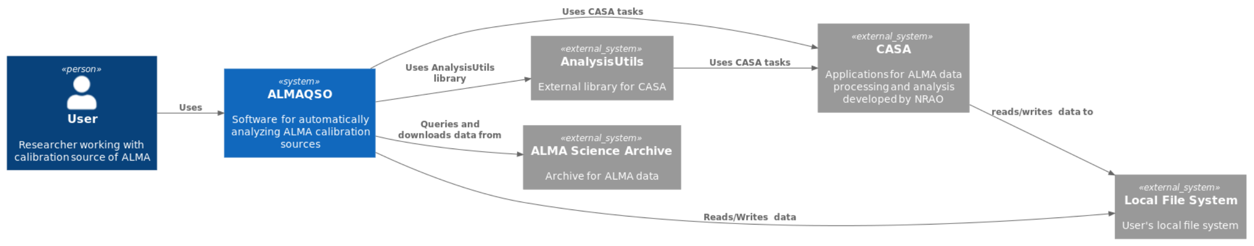

Fig. 1: C4-Context Diagram. (Web version: Click to show full-size diagram.)

3.2 Container

This section provides a overview of ALMAQSO showing the boundary of container (executable, library, database and so on) within the system. However, ALMAQSO is small project and consists of only one container, ALMAQSO Library.

Purpose

This diagram explains the high-level technology choices made within ALMAQSO and how responsibilities are distributed across the containers.

Audience

- Developers working on ALMAQSO
- New contributors

Components

Here, we explain only new components that were not explained in the System Context diagram.

- **ALMAQSO Library**: A Python library that provides functionalities for downloading, processing, and analyzing ALMA calibration source data. Python is widely used in the radio observational astronomy community, making it a suitable choice for this library.

C4-Container Diagram

3.3 Component

This section provides a detailed overview of the internal structure of the ALMAQSO Library container, breaking it down into its modules and classes. This section may be updated frequently since the development phase is moving from PoC (Proof of Concept) to a well-structured library.

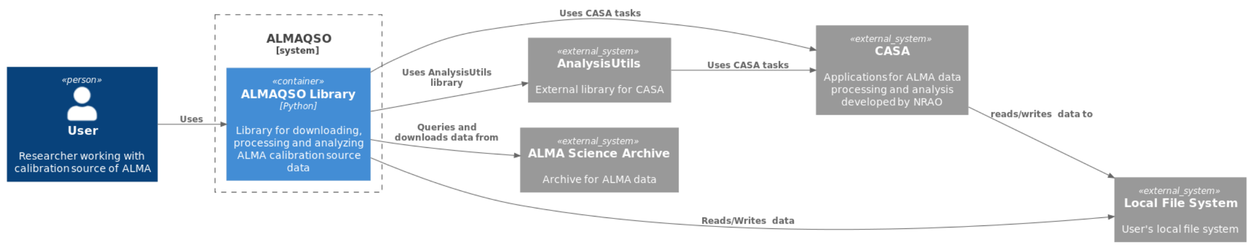

Fig. 2: C4-Container Diagram. (Web version: Click to show full-size diagram.)

Purpose

This diagram explains how the ALMAQSO Library is structured internally, showing the main components and their interactions.

Audience

- Developers working on ALMAQSO
- New contributors

Components

Here, we explain the main components of the ALMAQSO Library:

- **Almaqso Class**: The main class that provides the public API for users to interact with the library.
- **Logger Manager**: A component responsible for initializing and configuring logging within the library.
- **Query Class**: A class that handles querying the ALMA Science Archive for calibration source data.
- **Download Function**: A function that manages the downloading of data from the ALMA Science Archive.
- **Process Class**: A class that encapsulates the logic for processing downloaded data using CASA tasks.
- **Analysis Functions or Classes**: Components that provide functionalities for analyzing the processed data. This is not yet implemented.

C4-Component Diagram

4 API Reference

`class Almaqso`(*target: list[str]*, *band: int*, *cycle: str = ''*, *work_dir: str = './'*, *casapath: str = 'casa'*)

Bases: `object`

Public class for ALMAQSO. Users can call this class and its methods to use ALMAQSO's functionality.

Parameters

- **`json_filename`** (*`str`*) – JSON file name obtained from the ALMA Calibration Catalog.
- **`target`** (*`str`*) – Target source name.
- **`band`** (*`int`*) – Band number to work with.
- **`cycle`** (*`str, optional`*) – project name to work with. You can specify multiple cycles using , or ; and ~ like *"1,2"* (Cycle 1 and 2), *"1;2"* (Cycle 1 and 2), *"1~3"* (Cycle 1 to 3), *"1,4-6,8-10"* (Cycle 1, 4 to 6, 8 to 10), etc. Default is "" (all cycles).
- **`work_dir`** (*`str, optional`*) – Working directory. Default is './'.

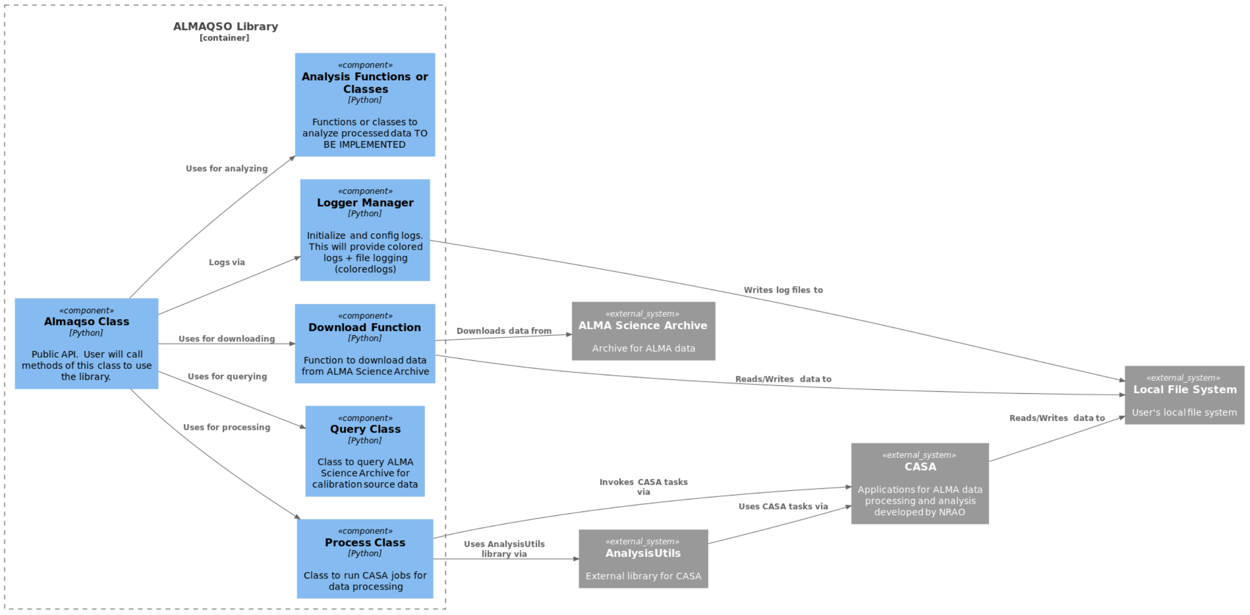

Fig. 3: C4-Component Diagram. (Web version: Click to show full-size diagram.)

- **`casapath`** (*`str, optional`*) – Path to the CASA executable. Default is 'casa'.

`process`(*n_parallel: int = 1*, *skip_previous_successful: bool = False*, *do_tclean: bool = False*, *tclean_mode: list[str] = ['mfs']*, *tclean_weightings: tuple[str, str] = ('natural', '')*, *do_selfcal: bool = False*, *kw_selfcal: dict[str, object] = {}*, *do_export_fits: bool = False*, *remove_casa_images: bool = False*, *remove_asdm: bool = False*, *remove_intermediate: bool = False*) → None

Download and process ALMA data.

Parameters

- **`n_parallel`** (*`int`*) – The number of the parallel execution.
- **`do_tclean`** (*`bool`*) – Perform tclean. Default is False.
- **`skip_previous_successful`** (*`bool`*) – Skip processing for previously successful tasks. Default is False.
- **`tclean_mode`** (*`list[str]`*) – List of imaging specmodes for tclean. "mfs" creates a MFS image, "mfs_spw" creates MFS images for each spw, and "cube" creates a cube image. Default is ["mfs"].
- **`tclean_weightings`** (*`tuple[str, str]`*) – Weighting scheme and robust parameter for tclean. Second element is the robust parameter for briggs weighting. Default is ("natural", "").
- **`do_selfcal`** (*`bool`*) – Perform self-calibration. Default is False.
- **`kw_selfcal`** (*`dict[str, object]`*) – Parameters for the self-calibration and *tclean* task.
- **`do_export_fits`** (*`bool`*) – Export the final image to FITS format. Default is False.
- **`remove_casa_images`** (*`bool`*) – Remove the CASA images after processing. This option only works if do_tclean is True. Default is False.
- **`remove_asdm`** (*`bool`*) – Remove the ASDM files after processing. Default is False.

- `remove_intermediate` (*bool*) – Remove the intermediate files after processing. Log of CASA will be retained. Default is False.

Returns
None