LBT-ADOPT
TECHNICAL REPORT

Doc.No : 481f301e
Version : 2.0
Date : 29 Oct. 2016

AOS
The Complete Guide

Luca Fini, Alfio Puglisi, Lorenzo Busoni, Francesco Tribioli

CAN: 481f301e

INAF - Osservatorio di Arcetri
`http://adopt.arcetri.astro.it/`

ABSTRACT

This report is a complete guide to the AOS both from the functional point of view and for many implementation aspects. It is intended to be useful in order to understand AOS functionality, so that instrument software programmers can better exploit AOS capabilities for their purposes, system programmers can be helped in troubleshooting or maintenance activities and telescope operators can better know how to use the AdOpt subsystem.
This report also includes all the information that was previously covered in document CAN 481f300 [1]

Revision history

Version	Date	CAN Number	Description
1.0	June 2009	481s301a	First draft
1.1	December 2010	481s301b	Pre commissioning version. Includes modifications and upgrades resulting from the tests in the Solar Tower Lab.
1.2	March 2011	481s301c	End of phase 1 commissioning version. Includes modifications and upgrades resulting from the commissioning activity at the telescope during commissioning of the FLAO unit N. 1 (right).
1.3	March 2012	481s301d	End of phase 2 commissioning version. Includes modifications and upgrades resulting from the commissioning activity at the telescope during commissioning of the FLAO unit N. 2 (left).
2.0	September 2016	481s301e	The new AOS version (16.x) includes support for ARGOS and for the FLAO Upgraded version (UAO). The architecture of the process has been largely modified.

Contents

Glossary of terms and acronyms

AdSec. Short for Adaptive Secondary Mirror. In this context usually refers to the group of *FLAO Supervisor* components controlling the hardware devices related to the secondary mirror.

ADAM. Ethernet controlled digital output. Used to enable various devices within the AdSec.

AdSec Server. The server running the *FLAO Supervisor* components which control the Adaptive Secondary hardware.

AdSec Arbitrator. The Adaptive Secondary Arbitrator. A component of the *FLAO Supervisor* which executes commands related to the *AdSec* coordinating the operations of the hardware devices in the Adaptive Secondary. Commands to *AdSec-Arb* may come either from a specific GUI or from *AO-Arb*.

AdSecArb. Short for AdSec Arbitrator.

ARGOS. (Advanced Rayleigh guided Ground layer adaptive Optics System) is the laser guide star and wavefront sensing facility added to the LBT AO system.

ARGOS Arbitrator. The process of the ARGOS software system implementing the communication interfaces with AOS.

FLAO System. The hardware and software components of the LBT first light Adaptive Optics System. Includes the Wavefront Sensor, the Adaptive Secondary Mirror, the *AO Servers* and several auxiliary devices (such as networking hardware) and includes the *FLAO Supervisor* and the real-time software.

FLAO Arbitrator. A component of the *FLAO Supervisor* which manages the execution of high-level commands, coordinating the operations of *WFS-Arb* and the *AdSec-Arb*. Commands to *AO-Arb* may come either from a specific interface or from *AOS*.

FLAO Servers. The servers running the *FLAO Supervisor* related processes.

FLAO Console. The operator console of either the *AdSec Server* or the *WFS Server*.

FLAO Supervisor. The software system which manages all the components of the *AO System*

AOArb. (Sometimes also referred to as FLAOArb) Short for FLAO Arbitrator.

AOSup (Sometimes also referred to as FLAOSup) Short for FLAO Supervisor.

BCU. Basic Control Unit. Electronics board used as basic building block for most of the electronics in the AO System (see [2]).

BCU 47. The BCU used as frame grabber for the CCD 47.

C-BCU. Crate BCU. The six BCUs which controls the AdSec.

CCD 39. The CCD used for the Wavefront sensor (also: *WFS CCD*).

CCD 47. The CCD used for the *TV*.

Copley. Motor driver for the Bayside stages.

Fastlink. The real-time data communication link between the WFS and the AdSec.

FLAO. First Light AO system for the LBT (or, if you like it better: FLorence AO System for LBT). The whole AO system for the LBT, including both hardware and software components.

Flowerpot. Auxiliary unit to control the calibration source and the related optics (cube beam splitter).

Hexapod. Mechanical support of the secondary mirror which allows to control the global mirror position with six degrees of freedom.

IIF. Instrument Interface, the TCS subsystem which provides a standard interface for instrument software.

MsgD. Message Dispatcher, the *FLAO Supervisor* message dispatching daemon.

OSS. Optical subsystem, the TCS subsystem which manages many devices in the telescope optical path. Most notably, from the point of view of the AO System, operates the hexapod.

RTDB. FLAO Real Time Database, the *FLAO Supervisor* own variable repository. Its functionalities are supported by `MsgD`.

S-BCU. Switch BCU. The BCU operating as input source switch for the AdSec.

TO. The Telescope Operator.

TTM. Tip-Tilt Mirror. A small mirror used to modulate the pupil image un top of the WFS pyramid.

TV. Technical Viewer. An auxiliary CCD camera used by the Wavefront Sensor to acquire the reference star.

WFS. The Wavefront Sensor. In this context usually refers to the software subsystem controlling the hardware devices related to the wavefront sensor.

WFS CCD. The CCD used in the FLAO *WFS* to measure the light wavefront deformation (also: *CCD 39*).

WFS Server. The server running the *FLAO Supervisor* components which control the Wavefront Sensor hardware.

WFS Arbitrator A component of the *FLAO Supervisor* which executes commands related to the *WFS* coordinating the operation of the hardware devices of the WFS. Commands to the WFS Arbitrator may come either from a specific GUI or from the *AOArb*.

WfsArb. Short for WFS Arbitrator.

1 Introduction

The complete LBT AO software system can be divided into three largely independent parts: 1) the AOS, 2) the FLAO Supervisor, 2) the ARGOS Supervisor. Figure 1 shows the relationships of the three parts.

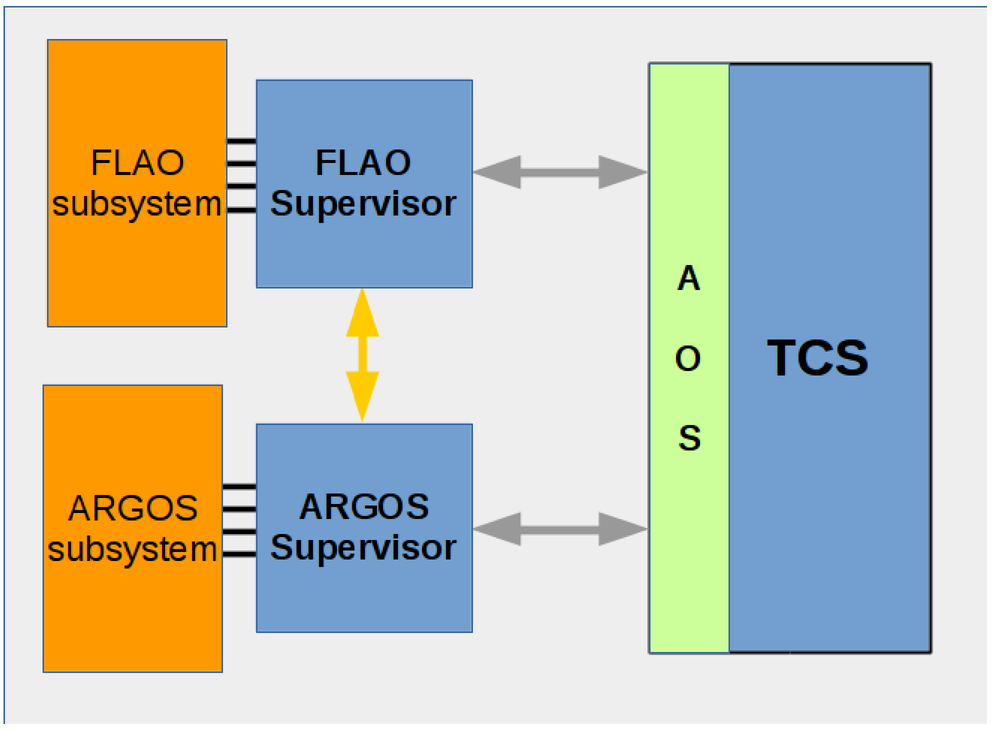


Figure 1: FLAO Supervisor architecture and interactions with TCS and ARGOS

1.1 The AOS

The Adaptive Optics Subsystem (AOS) is a standard software component of the LBT Telescope Control System (TCS) whose purpose is to interface the TCS to the two external subsystems: the FLAO Supervisor and the ARGOS Supervisor. It provides all the functionality needed for the interaction between the LBT Adaptive Optics system and the rest of the telescope, including instruments.

The AOS is essentially an interface layer between the TCS and the external subsystems. It defines a set of commands which can be used by other TCS subsystems[1] to operate the AO System during an observation and provides the update of several status variables from/to the TCS Data Dictionary and the external subsystems.

Although the main flux of control originates from the TCS and both the FLAOSup and the ARGOSArb operate mainly as slaves, AOS also allows the external subsystems to issue a number of commands

[1]The TCS subsystems currently using the AOS commands are the IIF and the AOSGUI, plus the test program `aosclient`.

to support required functionality, e.g.: status variable update, the "offload" mechanism and direct hexapod control (see: Sections 6.1 and 6.2).

1.2 The FLAO external subsystem

The Adaptive Optics subsystem (FLAO) is the unit providing natural guide star AO capabilities to LBT. The controlling software of the unit is referred to as FLAO-Supervisor in the following.

1.2.1 FLAO Supervisor Architecture

A detailed description of the software architecture of the FLAO Supervisor is provided elsewhere [3, 4], anyway, a brief general description is included here in order to provide information required to understand the operation of AOS.

Figure 2 shows in some details the overall architecture of the FLAOSup, and its relationships with the TCS and ARGOS.

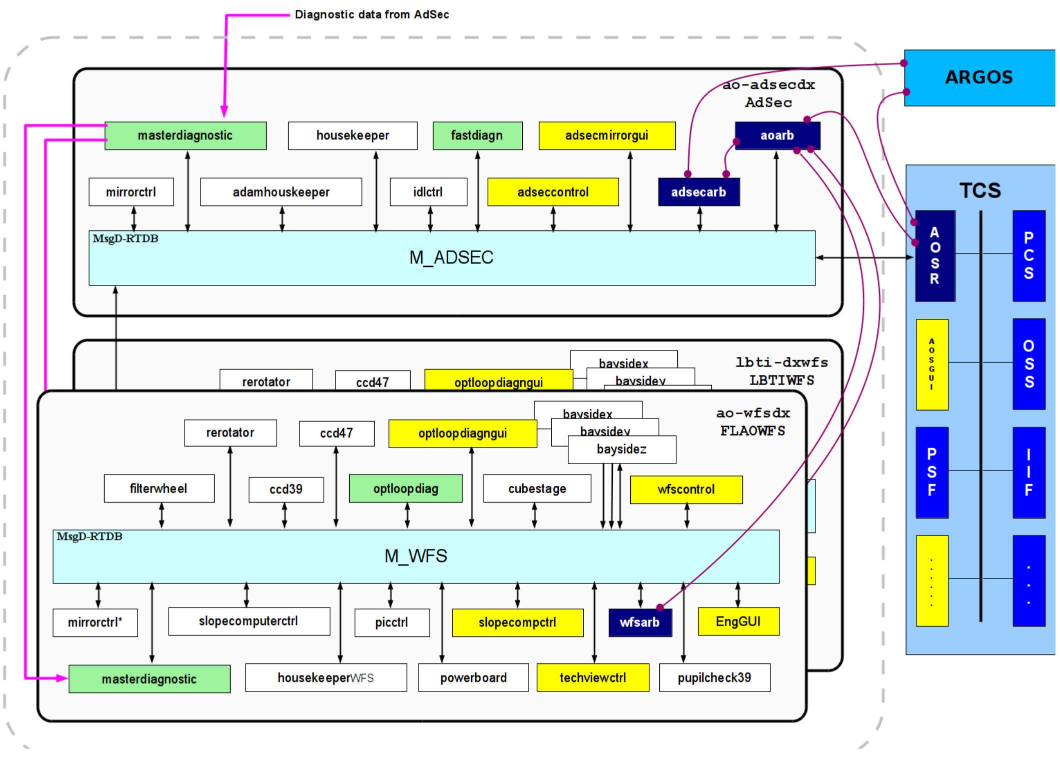


Figure 2: FLAO Supervisor architecture and interactions with TCS

The FLAOSup is based on a distributed architecture where several independent processes (indicated by rectangular boxes in the figure) cooperate in order to properly manage the underlying hardware, while ensuring safe operations of any device. The organization of processes in the FLAOSup reflects the splitting of FLAO system in three subsystems: the Adaptive Secondary (AdSec) and two Wavefront Sensors (FLAOWFS and LBTIWFS). The three software subsystems are currently hosted by three independent servers[2] indicated in the figure by the three large boxes named, respectively, `ao-adsecdx`, `ao-wfsdx` and `lbti-dxwfs`[3].

The two WFS servers run exactly the same code. The selection of what WFS subsystem is in use is done by a WFS specification argument when the AOS receives a preset command.

All interactions between FLAOSup processes are supported by a central process named MSG-RT DB[4] which allows the exchange of messages between any processes, supports a centralized repository of variables, and acts as a centralized logging facility. As shown in the picture several MSG-RT DB processes can communicate among each other to allow interaction between processes belonging to different subsystems. In our case the structure includes three interconnected MsgD-RTDB processes (named M_ADSEC, M_WFS and M_LBTI, respectively), one for each subsystem.

From the FLAOSup side, most of the interactions with TCS are managed by a dedicated process the "AO Arbitrator" (`aoarb`), whose main function is to receive commands from the AOS and coordinate the execution of requested operations by sending subcommands to the arbitrators controlling the WFS and the AdSec: `wfsarb` and `adsecarb`, respectively. The AO Arbitrator includes a finite state machine which defines the legitimate sequences of commands which can be issued. The main sequence of control is represented in figure 2 by the violet lines.

1.3 The ARGOS external subsystem

ARGOS is the unit providing artificial star based AO capabilities to LBT. The software controlling the ARGOS subsystem is referred to as ARGOS Supervisor in the following pages.

The ARGOS Supervisor has direct interactions with both the AOS and with the FLAO Supervisor. For the purposes of this report ARGOS/FLAO interactions are irrelevant and we will only deal with interactions between AOS and ARGOS in a dedicated section (see: Section 4).

During operation the ARGOS software system will receive commands from AOS and will send back status information and requests to the AOS much like the FLAO Supervisor.

[2]This hardware architecture was selected in order both to provide enough computing resources and to have a clear separation between the two components of the AO System, but it is not a requirement: the software design, in fact, allows different organization of processes, e.g.: the whole FLAOSup could run on a single server.

[3]The names refer to the right side AO System. For the left side the AO Servers are named `ao-adsecsx`, `ao-wfssx`, `lbti-sxwfs`, respectively.

[4]Message Dispatcher and Real-Time Database. See a more detailed description in [5].

2 AOS Architecture

Note: The current version of AOS is maintaining backward compatibility with the previous communication middleware just to be able, if necessary, to operate older version of the FLAO Supervisor. The related code will be removed as soon as the new ICE based FLAO Supervisor version will be released for science use. That part of the code is simply ignored in this report.

The AOS is a standard subsystem of TCS. As such it is globally structured like any other TCS subsystems. The functionality of AOS is essentially implemented in the `AOSSubsystem`, the `FLAOifApp` and in `ARGOSifApp`.

2.1 AOSSubsystem

`AOSSubsystem`[5] is structured as any other TCS subsystem and like other subsystems is executed in its own thread and is provided with an `execute()` method which is where part of the job is done.

In AOS, the `execute()` method provides for the initialization of the two main subprocess controlling the communication with external subsystem, namely: `FLAOifApp` for the FLAO subsystem and `ARGOSifApp` for the ARGOS subsystem and then enters a loop described in the following piece of pseudocode:

```
while Goon:
   FLAOifApp->sync()
   ARGOSifApp->sync()
   check_offload()
   sleep(800ms)
```

The `sync()` methods of `FLAOifApp` and `ARGOSifApp` have the purpose to keep updated the status of communication with the two external subsystems and to support the status variable exchange, for the part of it which is based on polling; `check_offload()` has the purpose to check whether a previous offload command is terminated[6].

The loop is executed at a convenient rate (typically 800 ms) to guarantee a good response to status changes.

The loop terminates when an explicit stop is requested.

AOS implements three communication paths: it receives commands from TCS subsystems (usually IFF and AOSGUI) as defined by `AOSClient`; it sends/receives commands and status info to/from the FLAO subsystem (see: `FLAOifApp` in Section 3); and sends/receives commands and status info to/from the ARGOS subsystem (see: `ARGOSifApp` in Section 4).

[5]AOSSubsystem code is in files: `AOS.cpp`, `AOS.hpp`.

[6]Each offload command may require several seconds to be executed, so subsequent offload requests are dropped until the last one is completed. The mechanism is detailed in Section 11.

2.2 TCS Command Interface

AOS defines a set of commands to be used by clients on the TCS side to operate the Adaptive Optics subsystems. The commands are divided into four groups:

1. **Internal commands**, which do not interact with external subsystems, but affect only AOS functioning.

   A list of defined internal commands can be found in table 1 and each command is described in details in Sections 5.1.1 through 5.1.5.

2. **AO commands**, which affect the currently active AO external subsystem. When using natural star AO (FLAO subsystem active), these commands operates the FLAO subsystem only; when doing laser star AO (ARGOS subsystem active), these commands are sent to the ARGOS subsystem. In this operating mode the FLAO subsystem is still operational but it is controlled directly by ARGOS except for a few housekeeping functions (see next point). The selection of currently active external subsystem is performed when receiving a `SetNewInstrument` command. See table 2. More detailed descriptions of each command can be found in Sections 5.2.1 through 5.2.13.

3. **FLAO commands**, housekeeping commands related only with FLAO (e.g.: switch on/off). See Section 5.3.

4. **ARGOS commands**, housekeeping commands related only with ARGOS. See Section 5.4.

A more detailed discussion of TCS commands can be found in Section 5.

2.3 External Command Interface

AOS provides support for the execution of commands issued by the external subsystems (either FLAO or ARGOS). External commands allow to update selected `DD` variables, to request mode offload to the TCS and to control directly the hexapod.

A complete description of external commands can be found in Section 6.

3 FLAOifApp

`FLAOifApp`[7] a.k.a.: FLAO Interface Application is the part of AOS dedicated to the communication with the FLAO subsystem.

It has several functions:

- Manages the communication with the FLAO subsystem detecting connections and disconnections and reacting properly.
- Sends commands to the FLAO external subsystem.
- Keep updated a number of TCS `DD` variables mirrored from the FLAO `RTDB`.
- Keep updated a number of FLAO `RTDB` variables mirrored from the TCS `DD`.
- Reacts to some commands issued from the FLAO subsystem. These include: hexapod movement requests, offload requests, telemetry data store requests.

`FLAOifApp` is launched in a dedicated thread by AOS during initialization as specified in Section 2.1, At start it performs the initialization of a few services, namely: 1) installation of command handlers to react to FLAO commands; 2) initialization of ICE middleware; 3) definition of variable lists for the two groups of variables to be mirrored.

After initialization `FLAOifApp` waits for any of the following events:

- AOSSubsystem calls any `FLAOifApp` command method
- One of the handlers receives a command/message from FLAO
- AOS calls the `sync()` method

3.1 Handlers

Handlers are special methods of `FLAOifApp` which manage asynchronous messages coming from the FLAOSup. They are part of the original communication middleware and were retained for compatibility purposes. They operate as callbacks to support commands issued by the external subsystems to be executed by TCS.

Handlers are installed as part of `FLAOifApp` initialization procedure. Each handler runs in its own thread and receives a predefined subset of the messages sent by `MsgD` to be processed.

Here follows a brief description of each handler.

[7]The related source code is in files: `FLAOifApp.cpp`, `FLAOifApp.hpp`, `FLAOCommands.cpp`, `FLAOCommands.hpp`.

3.1.1 `varnotify_hndl`

Receives notifications of relevant RTDB variables change.

The variable notification feature of the `RTDB` is used to mirror relevant variables from the `RTDB` into the TCS `DD`. As part of its initialization the `FLAOifApp` registers with the `RTDB` to be notified of updates of a specified set of variables.

Whenever any FLAOSup client writes to one of the variables this handler receives a notification (which includes the variable value). Upon notification, the handler writes the variable value into the corresponding variable in TCS `DD` and, possibly, performs some variable specific action.

A list of notified variables can be found in Section 7.

3.1.2 `hexapod_hndl`

This handler manages hexapod related commands[8] sent from the FLAOSup.

When an hexapod command is received, the handler checks whether the command can be accepted and, if positive, converts it into a command to OSS to be applied to the hexapod. An acknowledge message is sent back to the sender of the command.

Details on hexapod commands are covered in Section 6.1.1.

3.1.3 `housekeep_hndl`

This handler manages a set of housekeeping commands, mostly used for debugging and troubleshooting operations. For a list of defined housekeeping commands see Section 6.1.2.

This handler is obsolete and the related commands are no longer used; it will be removed in a future upgrade.

3.1.4 `offload_hndl`

This handler receives messages containing requests to offload errors accumulated in the adaptive mirror onto some other telescope device (e.g.: the pointing system, the hexapod or the primary mirror). A detailed description of the offload mechanism can be found in Section 11.

3.1.5 `default_hndl`

The default handler will receive all messages not trapped by other handlers. It simply logs warning messages if unexpected messages are received.

[8]In AOS design the capability to directly control the position of the adaptive mirror hexapod was included to support some specific operations needed for engineering or calibration procedures.

3.2 FLAOifApp::sync()

The `sync` method of FLAOifApp is called periodically by the main loop of AOS as specified in Section 2.1 in order to maintain the status of communication with the FLAO external subsystem, managing connection and disconnection as needed.

Here follows the sequence of operations performed every time the `sync` function is called:

- Checks the current status of `MsgD`. If it is connected:
  - Checks the status of AO Arbitrator[9]
  - Updates related status variables accordingly
  - Checks the status of ARGOS Arbitrator
  - Updates related status variables accordingly
  - Performs the mirroring of variables from the TCS `DD` to the `RTDB` by polling the variables from the `DD` and writing their values into the `RTDB`.
- If not connected:
  - Clears all status variables
  - Tries to reconnect

3.3 Interface to the FLAO Subsystem

`FLAOifApp` operates the FLAO subsystem by using an ICE based interface communicating with a single FLAO subsystem process: the AO Arbitrator[10].

[9] As AOS still supports also the older command interface with the AO Arbitrator the selection of the particular interface to be used is based on an internal flag corresponding to a configuration parameter.

[10] The interface is defined in files: `AODefinitions.ice` and `aoArb.ice`. Note that AOS only uses a subset of commands defined in the interface description file.

4 ARGOSifApp

`ARGOSifApp`[11] a.k.a.: ARGOS Interface Application, is the part of AOS dedicated to the communication with the ARGOS subsystem.

It has several functions:

- Manages the communication with the ARGOS subsystem detecting connections and disconnections and reacting properly.
- Sends commands to the ARGOS external subsystem.
- React to a number commands issued from the ARGOS subsystem. These include: hexapod movement requests, offload requests, status variable update requests.

Note: Variable updates for ARGOS is completely managed by the ARGOS subsystem via the command interface which provides methods to read and write TCS `DD` variables.

`ARGOSifApp` is launched in its own thread by AOS during initialization as specified in Section 2.1, At start it initializes the ICE middleware for the communication with the ARGOS subsystem software.

After initialization `ARGOSifApp` waits for any of the following events:

- AOS calls any `ARGOSifApp` command method
- The ARGOS external subsystem calls any `ARGOSifApp` callback
- AOS calls the `sync()` method

4.1 ARGOSifApp::sync()

The only purpose of `sync()` method is to manage the communication with the ARGOS external subsystem detecting connections and disconnections and reacting properly, more in details:

- On connection: Signal connection on
- On Disconnection: Signal connection off, clear ARGOS status variables

4.2 Interface to the ARGOS subsystem

All interactions between AOS and the ARGOS subsystems are supported by the ICE middleware. The interface[12] defines a set of commands which can be sent from AOS to ARGOS and a set of callbacks to receive commands from ARGOS.

[11] The related source code is in files: `ARGOSifApp.cpp`, `ARGOSifApp.hpp`.

[12] The interface definition is divided into a set of files: `ArgosAOS.ice`, `ArgosObservationArbitrator.ice`, `ArgosStates.ice`, `ArgosBeamAvoidance.ice`, `ArgosOperatorArbitrator.ice`, `ArgosTypes.ice`, `ArgosEngineeringArbitrator.ice`, `ArgosSnapshotable.ice`, in directory `.../aos/ice`.

4.2.1 Commands related to ARGOS subsystem

The set of commands which can be sent from AOS to the ARGOS subsystem are described in Section 5.4; commands issued by ARGOS for AOS are described in Section 6.2.

5 TCS Commands

AOS provides four sets of commands to allow TCS subsystems to control the Adaptive Optics subsystem: 1) Internal commands; 2) AO commands; 3) FLAO specific commands; 4) ARGOS specific commands.

Except for internal commands, the execution of a command will usually cause the sending of some command to one of the external subsystems (FLAO or ARGOS).

AOS is provided with a command locking mechanism to reject a command when a previous one is still executing if this could affect the operation. The locking mechanism, anyway, takes into account the fact that FLAO and ARGOS are independent subsystems, allowing some commands to operate concurrently.

Command locking[13] is based on the command groups as defined above. More in details:

1. *Internal commands* are executed unconditionally.
2. *FLAO commands* are rejected if another FLAO command is executing and if an AO Command is executing when the currently active external subsystem is FLAO.
3. *ARGOS commands* are rejected if another ARGOS command is executing and if an AO Command is executing when the currently active external subsystem is ARGOS.
4. *AO Commands* are rejected if another AO command is executing and if a FLAO[ARGOS] command is executing when the currently active external subsystem is FLAO[ARGOS].

All such commands are executed synchronously, i.e.: no other command can be executed before the previous one terminates. Each command has also a timeout after which it is considered as terminated with error. Timeout values are set up as configuration parameters (see: Section 12.1).

5.1 Internal commands

Internal commands are those affecting only AOS without any direct interaction with the external subsystems. They are listed in table 1.

5.1.1 `EnableARGOS/EnableFLAO`

These commands have the purpose to enable or disable the corresponding external subsystem.

They simply affect two `DD` boolean variables[14], which are checked by the communication loops of the two external subsystems. When the subsystem is disabled, no attempt would be made to open the communication with it. The result is as if the external subsystem was not operational. This allows to switch on and operate the external subsystem without affecting TCS, e.g.: when doing maintenance operations.

[13]The related code is in `AOS.cpp`, class `AOSSubsystem`, method `cmdStart()`.
[14]`aos[side].flao.enabled`, `aos[side].argos.enabled`

Table 1: Internal commands

TCS Command	Description	Sect.
`EnableARGOS`	Enable/disable communication with ARGOS	5.1.1
`EnableFLAO`	Enable/disable communication with FLAO	5.1.1
`BadSeeing`	Set/reset "bad seeing" flag	5.1.2
`ManAcquire`	Set/reset "manual acquire" flag	5.1.3
`RRMode`	Set/reset "bad seeing" flag	5.1.4
`SetLogging`	Enable/disable logging subsets at run-time	5.1.5

5.1.2 `BadSeeing`

Sets/resets the "bad seeing" flag. The command toggles the value of a `DD` boolean variable[15] (usually by means of the corresponding button on AOSGUI). The value of this variable is sent to the active external subsystem (FLAO or ARGOS) with the `PresetAO` command. The receiving subsystem will consider a `true` value as indication of "bad seeing" and modify the selection of AO parameters accordingly.

5.1.3 `ManAcquire`

Sets/resets the "manual acquire" flag. It toggles a `DD` boolean variable[16]. This affects the mode of operation of the `AcquireRefAO` command[17].

When the "manual acquire" mode is enabled, the `AcquireRefAO` command does not operate automatically but allows the interactive selection of the reference star (see description in Sect. 5.2.1).

5.1.4 `RRMode`

Sets/resets the "retroreflector mode" of operation. It toggles `DD` boolean variable[18]. When set AOS will modify the "offload modes" algorithm to respond properly to offload commands when the retroreflector is installed[19].

5.1.5 `SetLogging`

Enables/disables subsets of the logging messages at run-time; it can be used for debugging purposes.

This command is currently available only from the test client (`aosclient`, see Section B.1) and is called with an argument specifying which subset of logging messages must be enabled/disabled as follows:

[15] `aos[side].badSeeing`
[16] `aos[side].man_acquire`
[17] This capability is available only when operating with FLAO.
[18] `aos[side].rr_enabled`
[19] The same effect can be obtained by setting a corresponding variable in `RTDB`.

M/m:	Enable/disable logging of communication with FLAO `MsgD`
F/f:	Enable/disable logging of communication with FLAO Arbitrator
A/a:	Enable/disable logging of communication with ARGOS Arbitrator

5.2 AO commands

AO commands are those directly related with the Adaptive Optics operations. They will result in actual commands sent to the currently active external subsystem (either FLAO or ARGOS).

They are listed in table 2 and are described in details in the following sections.

Table 2: AO Commands

TCS Command	Description	Sect.
`AcquireRefAO`	Start the acquisition of reference star	5.2.1
`CheckRefAO`	Identify the reference star and return its position	5.2.2
`CorrectModes`	CorrectModes	5.2.3
`ModifyAO` [a]	Modify selected AO parameters	5.2.4
`OffsetXY` [b]	Offset telescope position	5.2.5
`OffsetXYg`	Offset telescope position	5.2.5
`OffsetZ`	Offset focus	5.2.6
`Pause`	Suspend the adaptive loop	5.2.7
`PresetAO` [c]	Set up external subsystem to be ready for reference star acquisition	5.2.8
`PresetAOg`	Set up external subsystem to be ready for reference star acquisition	5.2.8
`RefineAO` [a]	Optimize AO parameters	5.2.9
`Resume`	Resume the adaptive loop after a Pause	5.2.10
`SetNewInstrument` [d]	A new instrument has been authorized	5.2.11
`StartAO`	Close the adaptive loop	5.2.12
`Stop`	Stop current operation and go back to ready status	5.2.13

(a) This command has not been implemented and is reserved for future enhancements.
(b) This command is used by IFF. Its purpose is to compute the offset coordinates with respect to the current reference star specified as a *Position object*. Then the command `OffsetXYg` is called to request the actual offset. The latter command is also used directly by AOSGUI to specify an offset in millimeters in the focal plane (see description in Section 5.2.5).
(c) This command is used by IIF. It's purpose is to compute the reference star coordinates based on the *Position object* provided in the call. Then the command `PresetAOg` is called to request the actual operation. The latter command is also called directly by AOSGUI to request a preset specifying the reference star position in focal plane coordinates (see description is Section 5.2.8).
(d) The `SetNewInstrument` command has no counterpart in FLAO. The information related to currently authorized instrument is passed as argument to `PresetAO`. When ARGOS is active, instead, the corresponding routine of ARGOS interface is called.

Just for reference here follows the typical command sequence issued by IIF when starting a diffraction limited observation:

1. `Stop`: All command sequences begin with a "Stop"

2. `SetNewInstrument`: Only if a new instrument must be authorized.
3. `PresetAO`: Communicates details needed to preset the external subsystem
4. `AcquireRefAO`: Start acquisition of the reference star.
5. `StartAO`: Starts the AO loop[20].

5.2.1 `AcquireRefAO`

External commands used	
FLAO	ARGOS
`CheckRefAO`	`checkRefAO`
`AcquireRefAO`	`acquireRefAO`

The `AcquireRefAO` command is usually issued after a `PresetAO` in order to request the external subsystem to proceed to reference star acquisition.

The command usually calls the corresponding command of the active subsystem passing the list of arguments.

In either case the external subsystem, after acquisition will send back the computed AO loop parameters as detailed in table 3.

Table 3: `AcquireRefAO`/`CheckRefAO` command return values

Name	Type	Units	Comment
`aos[s].ao.param.dx`	real	mm	Pyramid X displacement with respect to nominal position
`aos[s].ao.param.dy`	real	mm	Pyramid Y displacement with respect to nominal position
`aos[s].ao.param.freq`	real	s	CCD Frequency (actually: period)
`aos[s].ao.param.gain`	real		Resulting loop gain
`aos[s].ao.param.nbins`	int		CCD binning
`aos[s].ao.param.nmodes`	int		Number of corrected modes
`aos[s].ao.param.filter1`	string		Selected position of filter wheel # 1
`aos[s].ao.param.filter2`	string		Selected position of filter wheel # 2
`aos[s].ao.param.slnull`	real		Selected slope null
`aos[s].ao.param.snmode`	real		Reserved for future use
`aos[s].ao.param.strehl`	real		Reserved for future use
`aos[s].ao.param.r0`	real		Measured R0
`aos[s].ao.param.ttmodul`	real		Tip-Tilt internal mirror modulation
TV Frame			Technical viewer frame

Operation details

[20]For the FLAO subsystem this command has actually no effect: the system is ready for diffraction limited already at the end of the previous `AcquireRefAO` command.

- **Interactive selection of reference object (FLAO subsystem only)**
  When the "interactive" mode is enabled (see: Section 5.1.3) the FLAO subsystem upon receiving the `AcquireRefAO` command will provide an image of the Technical Viewer camera to be displayed on the AOSGUI panel[21]. Then AOS waits for a "point and click" selection of the reference star from the AOSGUI and gets reference object coordinate values provided by the AOSGUI (in `DD` variable `aos[s].ref_xy`).

- **Reference object acquisition mode selection**
  The acquisition of the reference star can be performed in two ways: 1) by moving the WFS internal stages properly, or 2) by adjusting telescope pointing. The `AcquireRefAO` command is provided with an argument to select the desired acquisition method.

  Standard mode acquisition is performed by simply calling the corresponding external command (FLAO: `AcquireRefAO`, ARGOS: `acquireRefAO`).

  If the request is for "repointing" mode, the following sequence of operations is performed:

  1. A `CheckRefAO/checkRefAO` command is issued to external subsystem in order to evaluate the amount of XY offset needed.
  2. A command to adjust pointing is issued to PCS.
  3. A final `AcquireRefAO/acquireRefAO` command is sent to the external subsystem to actually perform reference star acquisition.

     Because of the repointing made at step 2, the following `AcquireRefAO` command is not expected to require the movement of WFS internal stages.

5.2.2 `CheckRefAO`

External command used	
FLAO	ARGOS
`CheckRefAO`	`checkRefAO`

The command is similar to `AcquireRefAO` in that it performs the same sequence of operations, except the final movement of FLAO stages to actually acquire the reference star.

After successful completion, the command returns the same parameters as the `AcquireRefAO` command (see table 3) where the first two item contain the position of the reference object. This can thus be used to measure the offset between the reference star nominal and actual positions.

An example of usage can be found in Section 5.2.1 where the "repointing" reference star acquisition mode is described.

[21]The technical viewer image is provided via the variable notification mechanism (see: Section 7).

5.2.3 `CorrectModes`

External command used	
FLAO	ARGOS
`CorrectModes`	`correctModes`

This command is used to apply a modal correction to mirror shape when the AO System is in Closed Loop (e.g.: to correct non common path aberrations). The command is obviously not available from the AOS GUI and can be issued only via the IIF.

A vector of Δ values must be specified as argument to the command.

AOS will directly issue the corresponding command to the currently active external subsystem and wait for completion.

5.2.4 `ModifyAO`

The `ModifyAO` command[22] has the purpose to support interactive adjustment of AO parameters. It may be used to request the external subsystem to modify the value of some AO loop parameter before closing the AO loop. The command must specify the set of AO loop parameters selected by the observer as detailed in table 4).

Table 4: *AOArb*:`ModifyAO` command input arguments

Name	**Type**	**Units**	**Comment**
`NModes`	int		Number of corrected modes
`Itime`	real	s	CCD integration time
`Nbins`	int		CCD binning
`TTMod`	real	TBD	Tip-Tilt internal mirror modulation
`F1spec`	string		Selected position of filter wheel # 1
`F2spec`	string		Selected position of filter wheel # 2

5.2.5 `OffsetXY/OffsetXYg`

External command used	
FLAO	ARGOS
`OffsetXY`	`offsetXY`

These are two flavors of the same command to request an offsetting of the telescope pointing.

[22]The `ModifyAO` command has been defined for possible future implementation, but the corresponding procedures have not yet been implemented in either FLAO or ARGOS.

- `OffsetXY` is usually called by IIF providing as argument a *Position Object* to define the reference star to be used. AOS retrieves from the *Position Object* the focal plane coordinates to be used and then calls `OffsetXYg`.
- `OffsetXYg` Sends a request to the active external subsystem to apply an offset, specifying the amount of offset in focal plane coordinates (millimeters). It is called by `OffsetXY` as specified above and can be called from the AOSGUI.

Notes:

1. In order to derive a sensible value for the command timeout, the latter is computed as a a linear function of the offset distance.

$$T_{out} = \texttt{FLAO\_OffsetXYTmo} + \texttt{FLAO\_OffsetSpeedFactOL} \times offset \qquad (1)$$

$$T_{out} = \texttt{FLAO\_OffsetXYTmo} + \texttt{FLAO\_OffsetSpeedFactCL} \times offset \qquad (2)$$

2. Due to the limited speed in the offload mechanism (see: Section 11), when the AO loop is closed the AO system can only perform very small offsets. When longer offsets are required the sequence `Pause`, `Offset`, `Offset` back, `Resume`, must be used.

5.2.6 `OffsetZ`

External command used	
FLAO	ARGOS
`OffsetZ`	`offsetZ`

This command is issued in order to offset the focus position of the active external subsystem by calling the corresponding command.

The command requires one delta value[23].

Note: In order to derive a sensible value for the command timeout, the latter is computed using a linear function of the offset distance as detailed in Note 1 to command `OffsetXY` above.

5.2.7 `Pause`

External command used	
FLAO	ARGOS
`Pause`	`pause`

[23]For the FLAO external subsystem the value is expressed in millimeters of Z-stage movement.

This command temporarily suspends the AO loop; the active external subsystem must remain ready to resume, i.e.: to close again the AO loop.

AOS will directly call the corresponding external subsystem command and wait for completion.

The command may be usually followed by either an `OffsetXY` or a `Resume` or a `Stop` command.

5.2.8 `PresetAO/PresetAOg`

External command used	
FLAO	ARGOS
`PresetAO`	`presetAO`

This command comes in two flavors:

- `PresetAO`: is called by IIF specifying the reference object as a *Position Object*. AOS will compute the corresponding focal plane coordinates from the *Position Object*, extract from it other required parameters (e.g.: magnitude) and then call `PresetAOg`.
- `PresetAOg`: is called specifying the reference star coordinates, and other parameters and will call the corresponding function of the currently active external subsystem. The purpose is to to prepare the external subsystem for a following `AcquireRefAO` command. `PresetAOg` can also be called directly from AOSGUI (see: Section 13).

Upon receiving the command the external subsystem will perform all the required preset operations needed to prepare for the acquisition.

When the `PresetAO` command has been completed and the telescope has reached the pointing position and is tracking and guiding on the target requested by the instrument, an `AcquireRefAO` or `CheckRefAO` command may follow.

5.2.9 `RefineAO`

External command used	
FLAO	ARGOS
`RefineAO`	`refineAO`

The `RefineAO` command[24] is defined to allow to request the external subsystem to try to perform a better estimation of the AO loop parameters. It should be issued after the `AcquireRefAO`.

[24]The `RefineAO` command has been defined for future development, but the corresponding procedures have not yet been implemented either in FLAO or in ARGOS.

Table 5: `PresetAOg` arguments

Name	Type	Units		Comment
`AOMode`	string		Req.	Either "FLAOTT" or "FLAOAO" or "TRUTH" or "ARGOSTT"
`wfsSpec`	string		Req.	Specifies the source of WFS data(e.g.: FLAO)
`focStation`	string		Req.	Specifies the focal station currently authorized (e.g.: `bentGregorianFront`)
`Instr`	string		Req.	Specifies the instrument currently authorized (e.g.: `IRTC`)
`SOCoords`	real[2]	mm	Opt.	Position of the scientific object in focal plane coordinates
`ROCoords`	real[2]	mm	Req.	Position of the reference object in focal plane coordinates
`Elevation`	real	radians	Req.	Telescope elevation [a]
`RotAngle`	real	radians	Req.	Angular position of rotator [a]
`GravAngle`	real	radians	Req.	Angular position of rotator with respect to gravity [a]
`Mag`	real		Opt.	Magnitude of reference star
`Color`	real		Opt.	Color Index of reference star
`R0`	real		Opt.	Estimated value of R0
`SkyBrghtn`	real		Opt.	Sky brightness
`WindSp`	real		Opt.	Wind speed
`WindDir`	real		Opt.	Wind direction

(a) This is the estimated value when the pointing position is reached. The value may be used by the external subsystem to select the proper look-up table and to preset the ADC.

5.2.10 `Resume`

External command used	
FLAO	ARGOS
`Resume`	`resume`

Resumes a suspended AO loop after a `Pause`. AOS will directly call the corresponding external system command and wait for completion.

The loop can be resumed successfully only if during the pause the telescope tracking (or guiding) system is able to maintain the correct pointing of the reference star; i.e.: for a limited amount of time.

5.2.11 `SetNewInstrument`

External command used	
FLAO	ARGOS
	`setNewInstrument`

This command is received whenever a new instrument is authorized. It is implemented differently for FLAO and ARGOS.

- FLAO has not a corresponding function: it was designed to get instrument information from the arguments of `PresetAO` command.
- ARGOS has a specific function to be notified of the authorization of a new instrument.

When receiving `SetNewInstrument` AOS will also select the currently active external subsystem: either FLAO or ARGOS.

5.2.12 `StartAO`

External command used	
FLAO	ARGOS
`StartAO`	`startAO`

The `StartAO` command is issued to start the Adaptive mode. The AOS will simply call the corresponding command of the currently active external subsystem and wait for a reply.

Note: The current implementation for FLAO subsystem is actually a no-op, because the AO loop is already closed at the end of a successful `AcquireRefAO` command.

5.2.13 `Stop`

External command used	
FLAO	ARGOS
`Stop`	`stop`

This command is issued to stop the current operation. AOS will directly call the corresponding command of the currently active external subsystem. The latter should cancel any setting defined by a previous `PresetAO` command and return to a state where it can accept another `PresetAO` command.

5.3 FLAO Commands

FLAO specific commands have been added to allow the T.O. to perform housekeeping operations on the FLAO subsystem. A list of command with a brief description can be found in table 6.

Upon receiving any command AOS will call the corresponding FLAO one (with the same name).

5.4 ARGOS Commands

ARGOS specific commands have been added to allow the T.O. to perform housekeeping operations on the ARGOS subsystem. A list of command with a brief description can be found in table 7.

Upon receiving any command AOS will call the corresponding ARGOS one (with the same name).

Table 6: FLAO specific commands

Command	Description
`AdsecOff`	Switch on adaptive secondary
`AdsecOn`	Switch off adaptive secondary
`AdsecRest`	Set adaptive secondary to rest state
`AdsecSet`	Set adaptive secondary to operating state
`PresetFlat`	Set a precalibrated mirror shape
`RefPosition`	Set acquisition coordinates (for manual acquire)
`SetZernikes`	Apply shape correction
`WfsOff`	Switch off FLAO or LBTI WFS
`WfsOn`	Switch on FLAO or LBTI WFS

Table 7: ARGOS specific commands

TCS Command	Description
`ArgosAcquire`	Acquire natural guide star and laser guide stars
`ArgosLaunch`	Propagate lasers
`ArgosMoveCalibrationSwingArmIn`	Operate ARGOS swing arm
`ArgosMoveCalibrationSwingArmOut`	Operate ARGOS swing arm
`ArgosMoveDichroicMirrorToParkingPosition`	Operate ARGOS dichroic
`ArgosMoveDichroicMirrorToWorkingPosition`	Operate ARGOS dichroic
`ArgosPowerLANOff`	Switch off launch subsystem
`ArgosPowerLANOn`	Switch on launch subsystem
`ArgosPowerLASOff`	Switch off lasers
`ArgosPowerLASOn`	Switch on lasers
`ArgosPowerLGSWOff`	Switch off laser WFS subsystem
`ArgosPowerLGSWOn`	Switch on laser WFS subsystem
`ArgosPowerOff`	Switch off all ARGOS subsystems
`ArgosPowerUp`	Switch on all ARGOS subsystems. First step of ARGOS complete setup
`ArgosPrepareForObservation`	Set up ARGOS subsystem to be ready for `PresetAO`. Fourth step of complete ARGOS setup
`ArgosSetUp`	Second step of complete ARGOS setup
`ArgosShutdown`	Shutdown the ARGOS subsystem
`ArgosStartUp`	Third step of complete ARGOS setup

6 External Commands

AOS implements a number of commands to be used by external subsystems (FLAO and ARGOS). Although FLAO and ARGOS perform very similar operations, because ARGOS support was designed year later than FLAO support, they are implemented very differently by AOS.

6.1 FLAO External Commands

FLAO external commands are implemented by means of the message handling mechanism provided by the `MsgD`, i.e.: they are not included in the ICE Interface.

The can be divided into four subgroups: Hexapod commands, Housekeeping commands, Variable Update commands and Offload commands.

6.1.1 FLAO Hexapod Commands

This set of commands are used by FLAO to control the secondary mirror hexapod. They are never used during an observation, but may be needed to perform engineering tasks, e.g.: for calibrations or maintenance operations.

Related command messages are received by a specific handler (`hexapod_hndl()` in file
tt FLAOifApp.cpp) and corresponding functions of AOSSubsystem are called.

Hexapod commands are listed in table 8 together with the corresponding AOSSubsystem commands and a brief description.

Table 8: FLAO Hexapod Commands

Command code	**AOSSubs. func.**	**Description**
HXPINIT	HXPinit	Initialize hexapod
HXPMOVETO	HXPmoveTo	Move hexapod to absolute position
HXPMOVEBY	HXPmoveRel	Move hexapod relative to current position
HXPMOVSPH	HXPmoveSphere	Move hexapod on a sphere
HXPNEWREF	HXPsetRelRef	Set new reference point
HXPGETPOS	HXPgetPos	Get hexapod current position
HXPGETABS	HXPgetAbs	Get hexapod absolute position
HXPOPENBRAKE	HXPopenBrake	Open brake
HXPCLOSEBRAKE	HXPcloseBrake	Close brake
HXPISINITIALIZED	HXPisInitialized	Test if hexapod has been initialized
HXPISMOVING	HXPisMoving	Test if hexapod is moving

6.1.2 FLAO Housekeeping Commands

This set of commands where designed to allow the FLAO external subsystem to affect some AOS behavior (e.g.: to modify logging verbosity on the fly). They have never actually been used by FLAO and are candidates for removal from a future version of AOS.

The corresponding handler in `FLAOifApp.cpp` is `housekeep_hndl`

They are listed in table 9.

Table 9: Housekeeping commands

Command	**Description**
LogItem	Requests to log some piece of information into the TCS Logging System.
LogOn	Activate mirroring to `MsgD` of AOS generated log Messages.
LogOff	Deactivate mirroring to `MsgD` of AOS generated log Messages.
DbgLevel	Set debug level of AOS

6.1.3 Variable Update Commands

This set of commands are used to notify AOS of the change of some FLAO internal variable. When AOS connects to the FLAO `MsgD`, it registers for notification on a set of variables[25] which are kept updated by FLAO issuing update commands.

Update command messages are received by a the handler (`varnotify_hndl()` in file `FLAOifApp.cpp`) and corresponding actions are performed: usually the action consists in updating a corresponding variable in TCS `DD`, in some cases a function is called to perform more complex tasks.

6.1.4 Offload Command

This is a single command used by FLAO to request the offload of accumulated shape errors to some telescope devices. Because it is a critical part of the AO functioning it is described in a dedicated section (see: Section 11).

6.2 ARGOS External Commands

ARGOS issued external commands are implemented as callbacks as specified by the ICE standard. They can be divided into four groups:

Telescope repoint commands, data dictionary access commands, hexapod commands, status commands.

[25] The code for FLAO variable support is in file `VarUtils.cpp`

6.2.1 Telescope Repoint Command

This includes a single command specifying a pointing offset to be requested to TCS.

The command internally calls the function `updateReferenceOrigin()` of PCS.

6.2.2 Data Dictionary Access Commands

This group includes a number of functions to read or write TCS Data Dictionary variables.

Implemented commands are listed in table 10.

Table 10: ARGOS Dictionary Access Commands

Command	**Description**
`getParameters`	Returns values of a set of `DD` variables [a]
`setParameters`	Sets the value of a set of `DD` variables [b]
`getDerotationAngleInRadian`	Returns the current value of the rotator angle
`getPointing`	Returns a complex structure containing values of several `DD` variables related with telescope pointing
`isAdSecSwingArmCompletelyDeployed`	Returns true if the AdSec swing arm is on place
`isAdSecSwingArmCompletelyRetracted`	Returns true if the AdSec swing arm is retracted
`isTelescopeGuiding`	Returns true if telescope is guiding
`isTelescopeTracking`	Returns true if telescope is tracking

(a) ARGOS has read access to any `DD` variable using this call.
(b) ARGOS has write access to only its own variables. I.e.: variables with the prefix: `aos.side[side].argos.`.

6.2.3 ARGOS Hexapod Commands

ARGOS hexapod commands are identical to FLAO hexapod commands. You may refer to Section 6.1.1 for the list and description of commands.

6.2.4 ARGOS Status Commands

ARGOS status commands are used by ARGOS to communicate relevant changes of status to AOS, usually with the purpose to show the status in the AOSGUI. Their effect is to update a corresponding `DD` variable. Commands are listed in table 11 together with the affected `DD` variable and the value assigned to it.

Table 11: ARGOS Status Commands

Command	**Affected DD Variable**	**Set to value**
onCalibrationSwingArmIn onCalibrationSwingArmOut	.argos.CalibrationSwingArmState	IN OUT
onDichroicToWorkingPosition onDichroicToParkingPosition	.argos.DichroicState	WORKING PARKING
onRemainingSatelliteClearanceDuration	.argos.SatelliteClearanceDurationInSecond	*Integer*
onSatellitePause onSatelliteClearance	.argos.SatelliteState	PAUSE CLEARANCE
onAircraftPause onAircraftClearance	.argos.AircraftState	PAUSE CLEARANCE
onBeamAvoidancePause onBeamAvoidanceClearance	argos.BeamAvoidanceState	PAUSE CLEARANCE
onPropagationShutterOpen onPropagationShutterClosed	.argos.PropagationShutterState	OPEN CLOSED
onArbitratorStateChange	.argos.ArbitratorState	*Status*
onLaserStateChange	.argos.LaserState	*Status*

7 Variables

AOS, alike all other TCS subsystems, uses a dedicated section in the TCS Data Dictionary to hold variables for various purposes. In the following tables the most relevant variables are grouped accordingly to their functions.

Notes regarding variable tables:

1. AOS variable names are shortened removing the common prefix: `aos.side[side]`.
2. Whenever the variable name contains the side indication the symbol x is used to represent either `L` or `R`.
3. A few variables used only for internal purposes are not listed.
4. For each variable it is indicated a generic source code, as follows:

   $\mathcal{A}$: the value comes from the ARGOS external subsystem;

   $\mathcal{C}$: the value comes from some configuration file;

   $\mathcal{F}$: the value comes from the FLAO external subsystem;

   $\mathcal{I}$: the variable is for internal use of AOS;

   $\mathcal{X}$: the variable comes from another TCS subsystem.
5. For a small subset of the variables it is important to asses that it is actually a "live" variable, i.e.: it is constantly updated by the owner process. Such variables (indicated with a "T"), have an associated timestamp variable; as part of the polling process, the timestamp variable is checked and if it becomes "older" than a preset expiration time the corresponding variable it is set to an undefined value. This allows the FLAOSup to take actions based on variable expiration.

7.1 TCS Variables used by FLAO

Table 12 lists `DD` variables from other TCS subsystems which are read from AOS and mirrored to the FLAO `MsgD`.

7.2 Functions of `DD` Variables Used by FLAO/ARGOS

A few values required by the external subsystems are functions of more than one `DD` variable (e.g.: `tel_tracking()` is a boolean value derived from the combination of three status variables: `mcs.azDrive.tracking`
`mcs.elDrive.trackingMode` and `mcs.onSource`). These functions are listed in table 13.

Note: the table shows the corresponding `MsgD` mirrored variable. ARGOS uses the same functions by polling via corresponding commands (see: Section 6.2).

Table 12: TCS variables mirrored to FLAO Supervisor

DD **variable**		FLAO MsgD variable
env.weather.lbt.windDirection	$\mathcal{X}$	AOS.EXTERN.WINDDIRECTION
env.weather.lbt.windSpeed	$\mathcal{X}$	AOS.EXTERN.WINDSPEED
gcs.side[side].GuideCam.centroid_FWHM_X	$\mathcal{X}$	AOS.x.GUIDECAM.CENTROID.X
gcs.side[side].GuideCam.centroid_FWHM_Y	$\mathcal{X}$	AOS.x.GUIDECAM.CENTROID.Y
iif.DIMM.seeing	$\mathcal{X}$	AOS.DIMM.SEEING
mcs.azDrive.position (T)	$\mathcal{X}$	AOS.TEL.AZ
mcs.elDrive.position (T)	$\mathcal{X}$	AOS.TEL.EL
mcs.rotatorSide[side].rotators[n].actualPositionASec	$\mathcal{X}$	AOS.x.ROTATOR.ANGLE
oss.side[side].adsc.abs_pos[6]	$\mathcal{X}$	AOS.x.HEXAPOD.ABS_POS
oss.side[side].terc.abs_pos[4]	$\mathcal{X}$	AOS.x.TERTIARY.ABS_POS
pcs.pointingStatus.achieved.achieved_DEC.Radians	$\mathcal{X}$	AOS.TEL.DEC
pcs.pointingStatus.achieved.achieved_RA.Radians	$\mathcal{X}$	AOS.TEL.RA

Table 13: Functions of DD variables

Function		MsgD variable	Description
hbs_on()	$\mathcal{I}$	AOS.TEL.HBS_ON	True when the HBS system is on
hexapod_status()	$\mathcal{I}$	AOS.x.HEXAPOD.STATUS	Hexapod status
swa_atStop()	$\mathcal{I}$	AOS.x.SWA.DEPLOYED	True when the swing arm is deployed
tel_guiding()	$\mathcal{I}$	AOS.TEL.ISGUIDING	True when the telescope is guiding
tel_tracking()	$\mathcal{I}$	AOS.TEL.ISTRACKING	True when the telescope is tracking
vent_on()	$\mathcal{I}$	AOS.x.TEL.VENT_ON	True when the ventilation system is on

7.3 Variables Related with the FLAO Adaptive Secondary

A number of variables are related to the status of the Adaptive Sedcondary. They are listed in table 14.

7.4 Variables related with the Status of the AO Loop

Variables listed in Tables 15 and 16 are related to the AO loop. In particular those listed in Table 16 contain specific AO parameters values updated from return values afters calling commands: AcquireRefAO, CheckRefAO and ModifyAO.

7.5 FLAO External Subsystem Variables

The variables listed in Table 17 are generic parameters related to the FLAO external subsystem. More parameters specific for the wavefront sensor of the FLAO subsystem are described in Section 7.6.

7.6 FLAO WFS and LBTI WFS Specific Variables

FLAO WFS and LBTI WFS specific variables are listed in Table 18. The two sets of variables are identical.

Table 14: FLAO AdSec Variables

`.adsec.act_failed`	$\mathcal{F}$		Failing actuator number (for AOSGUI)
`.adsec.anem_speed`	$\mathcal{F}$	ANEM.x.SPEED	Current anemometer values [a]
`.adsec.anem_time`	$\mathcal{I}$		Timestamp of last anemometer data update
`.adsec.anem_upd`	$\mathcal{F}$		True when anemomenter data are regularly updated
`.adsec.coil_status`	$\mathcal{F}$		Coil status indicator (for AOSGUI)
`.adsec.contamination`	$\mathcal{F}$		Contamination alarm status (for AOSGUI)
`.adsec.do_offload`	$\mathcal{F}$		Offload enable flag [b]
`.adsec.elev_upd`	$\mathcal{F}$		Elevation data update is OK [c]
`.adsec.health`	$\mathcal{F}$		AdSec is ready for operations
`.adsec.pwstatus`	$\mathcal{F}$		AdSec global status: off,safe,set
`.adsec.msg`	$\mathcal{F}$		Generic message from AdSec (to be displayed as Event)
`.adsec.nwact`	$\mathcal{F}$		Not working actuator indicator [d] (for AOSGUI)
`.adsec.offload`	$\mathcal{F}$		Last offload vector
`.adsec.safeskip_perc`	$\mathcal{F}$		Safe Skip Frame percentage (for AOSGUI)
`.adsec.shape`	$\mathcal{F}$		Current AdSec shape name
`.adsec.status`	$\mathcal{F}$		Adsec status string (for AOSGUI)
`.adsec.tss_status`	$\mathcal{F}$		AdSec TSS status (for AOSGUI)

(a) Values are: (X,Y,Z,Module) at 1 second time average, (X,Y,Z,Module) at 10 seconds average, (X,Y,Z,Module) at 60 seconds average.
(b) This variable is controlled by the `FastDiagnostic` process. When set the offload mechanism is activated (see: Section 11).
(c) This variable is set when the FLAO subsystem receives continuous elevation updates from TCS at the expected rate.
(d) This variable is only required for backward compatibility with pre-UAO FLAOSup.

Table 15: Generic AO loop variables

`.ao.ao_ready`	$\mathcal{F}$	FLAO is ready for diffraction limited operations
`.ao.correctedmodes`	$\mathcal{F}$	Current number of corrected modes (for AOSGUI)
`.ao.loopon`	$\mathcal{F}$	AO loop closed indicator (for AOSGUI)
`.ao.mode`	$\mathcal{F}$	Current AO mode (for AOSGUI)
`.ao.msg`	$\mathcal{F}$	Generic message from AO Arbitrator (to be displayed as Event)
`.ao.sl_ready`	$\mathcal{F}$	FLAO is ready for seeing limited operations
`.ao.status`	$\mathcal{F}$	AO Arbitrator FSM status
`.ao.strehl`	$\mathcal{F}$	Loop quality indicator [a]
`.ao.wfs_source`	$\mathcal{F}$	Current WFS in use (either FLAO or LBTI)

(a) This variable is not currently used.

7.7 ARGOS External Subsystem Variables

ARGOS specific variables are listed in table 19. Most of these variable are directly updated by the ARGOS subsystem (see also Section 6.2).

Table 16: AO loop specific parameters

`.ao.param.dx`	$\mathcal{F}/\mathcal{A}$	Reference object position (X)
`.ao.param.dy`	$\mathcal{F}/\mathcal{A}$	Reference object position (Y)
`.ao.param.filter1`	$\mathcal{F}/\mathcal{A}$	Selected filter 1
`.ao.param.filter2`	$\mathcal{F}/\mathcal{A}$	Selected filter 2
`.ao.param.freq`	$\mathcal{F}/\mathcal{A}$	WFS CCD frequency
`.ao.param.gain`	$\mathcal{F}/\mathcal{A}$	AO loop gain
`.ao.param.mag`	$\mathcal{F}/\mathcal{A}$	Measured magnitude of reference star
`.ao.param.nbins`	$\mathcal{F}/\mathcal{A}$	WFS CCD binning
`.ao.param.nmodes`	$\mathcal{F}/\mathcal{A}$	Number of corrected modes
`.ao.param.r0`	$\mathcal{F}/\mathcal{A}$	Measured R0 value
`.ao.param.slnull`	$\mathcal{F}/\mathcal{A}$	Slope null indicator
`.ao.param.snmode`	$\mathcal{F}/\mathcal{A}$	Reserved for future use
`.ao.param.strehl`	$\mathcal{F}/\mathcal{A}$	Currently unused
`.ao.param.ttmodul`	$\mathcal{F}/\mathcal{A}$	Tip-Tilt modulation amplitude

Table 17: FLAO external subsystem specific variables

`.flao.arb_connected`	$\mathcal{I}$	Flag indicating that AO Arbitrator is communicating
`.flao.arb_ctime`	$\mathcal{I}$	Arbitrator connection time
`.flao.arb_endpoint`	$\mathcal{I}$	Arbitrator ICE endpoint
`.flao.arb_ident`	$\mathcal{I}$	Arbitrator version
`.flao.enabled`	$\mathcal{I}$	FLAO communication enabled flag
`.flao.ice_if`	$\mathcal{I}$	FLAO communication enabled flag
`.flao.idlstat`	$\mathcal{F}$	Flag indicating IDL server status
`.flao.intv_mode`	$\mathcal{I}$	Flag indicating Intervention mode
`.flao.labmode`	$\mathcal{F}$	Flag indicating that lab mode is active
`.flao.log_arbitrator`	$\mathcal{F}$	Flag indicating that logging of arbitrator communication is active
`.flao.log_msgd`	$\mathcal{F}$	Flag indicating that logging of `MsgD` communication is active
`.flao.loopclosed`	$\mathcal{I}$	Time of last loop close command
`.flao.loopopened`	$\mathcal{I}$	Time of last loop open command
`.flao.msgd_connected`	$\mathcal{I}$	Flag indicating that MsgD is communicating
`.flao.msgd_ctime`	$\mathcal{I}$	MsgD connection time
`.flao.msgd_ident`	$\mathcal{I}$	MsgD identity string
`.flao.msgd_ip`	$\mathcal{I}$	MsgD IP address
`.flao.msgd_port`	$\mathcal{I}$	MsgD IP port
`.flao.rotindex`	$\mathcal{I}$	Flao rotator index
`.flao.rr_enabled`	$\mathcal{I}$	Flag indicating “retroreflector” mode
`.flao.synctimeout`	$\mathcal{I}$	FLAO synchronization timeout (millisec)

7.8 Miscellaneous Variables

In Table 20 a set of variables for miscellaneous uses are listed.

Table 18: FLAO WFS and LBTI WFS specific variables

`.flaow[.lbtiw].active`	$\mathcal{I}$		Flag indicating that this WFS is currently active
`.flaow[.lbtiw].ccdbin`	$\mathcal{F}$	`ccd39.`x`.XBIN.CUR`	Current WFS CCD binning
`.flaow[.lbtiw].ccdfreq`	$\mathcal{F}$	`ccd39.`x`.FRMRT.CUR`	Current WFS CCD frequency (actually: period in seconds)
`.flaow[.lbtiw].connected`	$\mathcal{I}$		Flag indicating that the WFS subsystem is communicating
`.flaow[.lbtiw].counts`	$\mathcal{F}$	`optloopdiag.`x`.COUNTS`	Current CCD counts
`.flaow[.lbtiw].filter1`	$\mathcal{F}$	`filterwheel1.`x`.POSNAME.CUR`	Current CCD counts
`.flaow[.lbtiw].health`	$\mathcal{F}$		Flag indicating that the WFS subsystem is ready for operations
`.flaow[.lbtiw].mod_ampl`	$\mathcal{F}$	`wfsarb.`x`.MOD_AMPL`	Pyramid modulation amplitude (for AOSGUI)
`.flaow[.lbtiw].msg`	$\mathcal{F}$	`wfsarb.`x`.MSG`	Generic message from WFS (to be displayed as Event)
`.flaow[.lbtiw].no_subaps`	$\mathcal{F}$	`slopecompctl.`x`.NSUBAPS.CUR`	Number of sub apertures (for AOSGUI)
`.flaow[.lbtiw].pwstatus`	$\mathcal{F}$		Global status (ON, OFF)
`.flaow[.lbtiw].status`	$\mathcal{F}$	`wfsarb.`x`.FSM_STATE`	WFS arbitrator status
`.flaow[.lbtiw].tv_angle`	$\mathcal{I}$		AOSGUI compass angle (currently unused)
`.flaow[.lbtiw].tv_filename0`	$\mathcal{I}$		
`.flaow[.lbtiw].tv_filename1`	$\mathcal{I}$		

8 Events and Logging

`AOS` is provided with 5 levels of log messages[26], in decreasing order of verbosity: `trace`, `debug`, `info`, `warning` and `error`.

The first two levels (most verbose) correspond only to lines logged via the *Syslog* mechanism, are usually disabled and can be enabled either by a configuration parameter (see: Section 12.1), or by means of the SetLogging command (see: Section 5.1.5).

The following three levels correspond both to *Syslog* messages and to TCS Events and thus are also logged to the LSS system. `Warning` and `error` events are obviously related to error conditions of increasing severity, while `info` events are generated to log command execution and offload requests.

9 Telemetry data

The AOS stores a selection of AO Supervisor data into the TCS telemetry store.

[26]The related source code is in files: `AOSEvent.hpp`, `AOSEvent.cpp`.

Table 19: ARGOS specific variables

.argos.AircraftState	$\mathcal{A}$	Aircraft state
.argos.ArbitratorState	$\mathcal{A}$	Arbitrator state
.argos.BeamAvoidanceState	$\mathcal{A}$	Beam avoidance state
.argos.CalibrationSwingArmState	$\mathcal{A}$	Calibration swing arm state
.argos.DichroicIn	$\mathcal{A}$	Dichroic is in optical path
.argos.DichroicState	$\mathcal{A}$	Dichroic state
.argos.LaserState	$\mathcal{A}$	Laser state
.argos.LGSWState	$\mathcal{A}$	Laser guide star WFRS state
.argos.PropagationShutterState	$\mathcal{A}$	Propagation shutter state
.argos.SatelliteClearanceDurationInSecond	$\mathcal{A}$	Satellite clearance duration (in sec)
.argos.SatelliteNextCloseDurationInSecond	$\mathcal{A}$	Satellite next clearance duration (in sec)
.argos.SatelliteState	$\mathcal{A}$	Satellite state
.argos.connected	$\mathcal{I}$	Flag indicating that the external subsystem is communicating
.argos.ctime	$\mathcal{I}$	ARGOS conenction time
.argos.enabled	$\mathcal{I}$	Communication enabled flag
.argos.ident	$\mathcal{I}$	ARGOS arbitrator identification string
.argos.log_arbitrator	$\mathcal{I}$	Flag indicating that logging of arbitrator communication is active
.argos.obsendpoint	$\mathcal{C}$	ARGOS Observation interface endpoint
.argos.operendpoint	$\mathcal{C}$	ARGOS Operator interface endpoint
.argos.rotindex	$\mathcal{C}$	ARGOS rotator index
.argos.severity	$\mathcal{A}$	Currently unused
.argos.synctimeout	$\mathcal{C}$	ARGOS synchronization timeout (millisec)

Table 20: Miscellaneous variables

.arbitrator	$\mathcal{I}$	Currently enabled arbitrator
.badSeeing	$\mathcal{I}$	Bad seeing flag
.flipX	$\mathcal{C}$	Bad seeing flag
.man_acquire	$\mathcal{I}$	Manual reference object selection flag
.oflm_gain[22]	$\mathcal{C}$	Offload vector coefficients
.oflm_threshold[22]	$\mathcal{C}$	Offload vector thresholds
.tel_enabled	$\mathcal{I}$	Telemetry collect enabled flag
.verbosity	$\mathcal{I}$	Bad seeing flag
.wfs_select	$\mathcal{I}$	Currently selected WFS (either FLAO or LBTI)
.zern_gain[22]	$\mathcal{C}$	Calibration coefficients for SetZernikes command

A list of data items stored into telemetry is shown in table 21.

Table 21: Data items stored into telemetry system

Group: **AdSec offload command**			
Name	Type	Rate	**Description**
z00 . . . z21	Float	N.A.	Mode offload Zernike coefficients. Stored whenever a correction is requested
cmd status	Int	N.A.	Offload command request status: Bitwise or of the values: 1: tip/tilt/focus request; 2: higher order request; 4: request not applied.

Group: **Anemometer speed**			
Name	Type	Rate	**Description**
1s Avg	Float	1 Hz	1 second running mean
10s Avg	Float	1 Hz	10 seconds running mean
1m Avg	Float	1 Hz	1 minute running mean

10 TV Image Frames

The AOS GUI (see: Section 13) includes the capability to display images from the Wavefront Sensor Technical Viewer. Such images are of two types: a) frames continuously generated by the TV CCD during its operation, and b) the specific frame when the reference source has been acquired as a consequence of an `AcquireRefAO` command. The former are transmitted asynchronously by means of the variable notification mechanism supported by the `MsgD`; the latter are sent in the set of parameters returned by the `AcquireRefAO` command. The two types of images are displayed independently.

Technical Viewer images are formatted as specified in the following structure:

```
struct TVIMAGE {
     int rows;       // Number of rows
     int cols;       // Number of cols
     unsigned char pix[];
}
```

Pixels are represented as gray levels in the range 0-255. The structure is packed into a byte stream of the proper length.

Due to implementation details of the TCS middleware, it is not practical to use either the parameter passing mechanism of TCS commands or the `DD` to transmit even moderately large data sets such as the 256x256 image from the TV.

For this reason an indirect mechanism is used to send TV frames to be displayed on the AOS GUI: frames from the TV are written to a temporary file in a NFS shared directory as soon as they are received from FLAOSup. AOS GUI polls for changes in file access date and displays the file content whenever the date changes.

The paths for TV image support files are specified in a configuration parameter.

11 Offload Modes

Mode offload is needed during closed loop when the AdSec gets near to some physical limit, in order to offload the error on the first modes accumulated in the deformable mirror to some other telescope device, typically the hexapod (for tip-tilt and focus) and the primary (for higher order components).

A typical example is when the telescope accumulates tracking errors: if the adaptive loop is on, those errors are compensated by the adaptive secondary by an increasing static tip-tilt component. As errors increase the AdSec would finally get close to intrinsic limits in the maximum amount of tip-til it can compensate. The error must thus be lowered, e.g.: by adjusting the telescope pointing or by moving the hexapod.

The mode offload mechanism is based on specific commands sent by FLAOSup and received by the related handler[27]. The handler forwards the request, consisting in an array of 22 zernike terms, to AOSSubsystem.

The related method in AOSSubsystem[28] processes the requests as follows:

1. The offload array is divided into two parts: the low order part (elements 2 to 4: the first element is never considered) related to tip/tilt and focus terms, and the high order part, including elements from 5 to 22. The two parts are independently compared against an array of threshold values[29].

   If the absolute value of any of the low order offload components is greater than the corresponding threshold the `applyTTFolload()` function is called. Analogously the `applyHOoffload()` function is called if the absolute value of any of the high order component is greater than the corresponding threshold.

2. Upon the call each function checks whether a previous correction is still being executed. If this is the case the correction request is simply dropped,[30] otherwise the offload array is multiplied by an array of gains[31] to generate a proper correction command to be sent to the PSF subsystem: `applyTTFoffload()` sends a `secondarymirror.setCollShellOffload` command, while `applyHOoffloads()` sends a `setZernikes` command.

3. Because the two functions affect independent telescope parts, i.e.: : the hexapod for Tip/Tilt/Focus and the primary for the higher order corrections, they can be executed concurrently. In order to allow concurrent execution the corresponding TCS commands are launched asynchronously and the function is locked to block any following command. The termination of the currently active commands is checked periodically[32] so that when the correction command terminates the function lock is released to allow for a following offload command.

[27]An ICE interface and the related callback to receive offload commands has been defined for the support of ARGOS. The current implementation, anyway, is still based on the mechanism described here, even when the ARGOS subsystem is active.

[28]Code is in file `AOS.cpp`, class `AOSSubsystem`, method: `offload()`

[29]Threshold values are defined in the AOS configuration file (see: Section 12).

[30]The External subsystems are requested to send offload messages at a rate not greater than one Hertz. Despite this, the actual offload execution by the TCS may be slower than that. Discarding offload request while the previous one is being served, ensures that the TCS does not receive more offload commands than it can manage.

[31]Gain values are defined in the AOS configuration file (see: Section 12).

[32]See also Section 2.1.

The above sequence is modified by the status of two variables:

`aos.side[s].adsec.do_offload`: if zero, no offload correction is applied. The variable is controlled by the FLAOSup to disable offload of modes when needed.

`aos.side[s].rr_enabled`: if non zero (1) indicates "retroreflector mode". In this mode only the low order offload correction is applied. The variable must be set to 1 when the retroreflector is installed. It can be set either from the FLAOSup side or by using the RRMode command (see: Section 5.1.4).

12 Configuration Items

Configuration data for AOS consists in a general configuration file: `.../aos/etc/aos.conf` and in a few data files stored in directory `.../aos/configuration/AOS`.

12.1 AOS General Configuration

The general configuration file contains all run time parameters needed for running AOS. The file format is the same used for the other TCS subsystems. AOS general configuration parameters are listed in table 22 with a brief explanation.

12.2 AOS Configuration Data Files

AOS uses a few configuration data files stored in directory: `.../aos/configuration/AOS`.

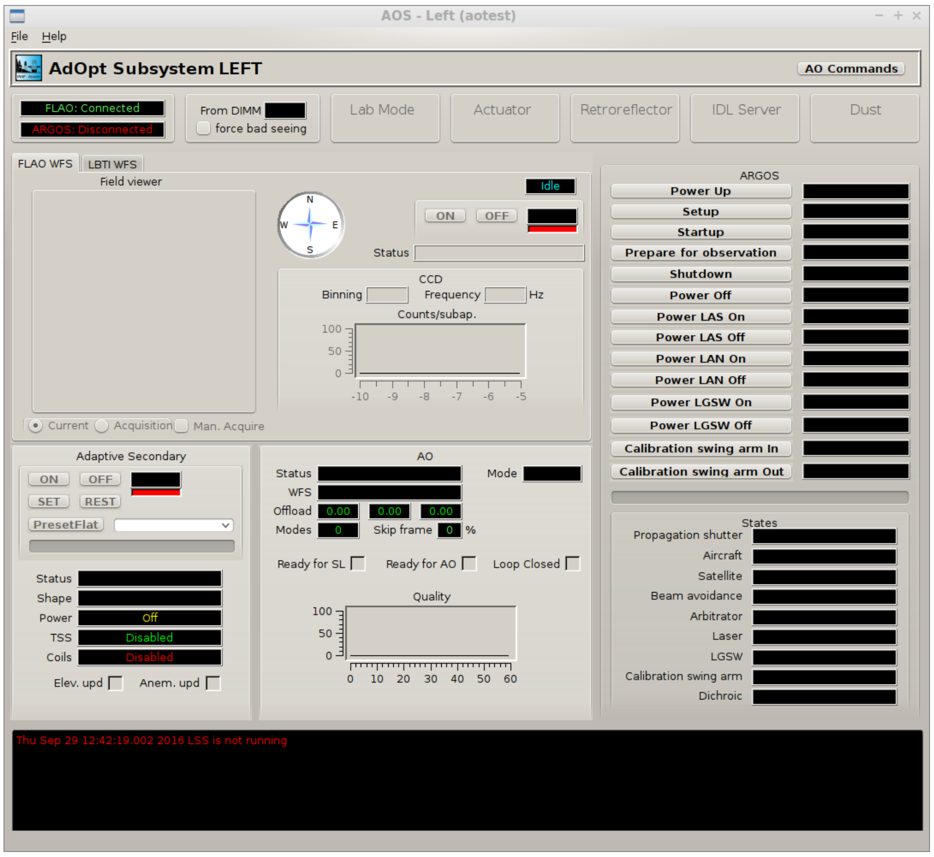
- `ARGOSDXOffloadModes.dat`: Threshold and gain arrays for Mode Offload when ARGOS external subsystem is active, right side.
- `ARGOSSXOffloadModes.dat`: As above for left side.
- `FLAODXOffloadModes.dat`: Threshold and gain arrays for Mode Offload when FLAO external subsystem is active, right side.
- `FLAOSXOffloadModes.dat`: As above for left side.
- `DXZernikes.dat`: Gains array for SetZernikes command, right side.
- `SXZernikes.dat`: As above for left side.

Table 22: AOS General Configuration Items

`AOSFlats` (map)	List of *instrument* to *flat name* associations
`FLAO_OffsetSpeedFactOL`	Open loop speed factor to compute OffsetXY and OffseZ commands timeouts (see: Section 5.2.5)
FLAO_OffsetSpeedFactCL	As above for closed loop
FLAO_*xxx*Tmo	A set of 20 parameters specifying timeouts for FLAO specific commands
ARGOS_*xxx*Tmo	A set of 38 parameters specifying timeouts for ARGOS specific commands
AOS*x*verbosity	Define verbosity for messages sent to syslog (0,1,2)
AOS*x*MsgDIp	IP address of FLAO `MsgD` (AdSec server IP)
AOS*x*MsgDPort	IP port number of FLAO `MsgD`
AOS*x*flaoICEifEP	ICE end point for FLAO communication. If left blank, AOS will use the older communication protocol.
AOS*x*tv_file	Full path of technical viewer image files (see: Section 10)
AOS*x*fl_file	Full path for flat list file
AOS*x*simulation	If *true* AOS will start in simulation mode
AOS*x*telEnable	If *true* telemetry sotre is enabled
AOS*x*rotatorIndex	Index of rotator used for FLAO (0:front, 1:center, 2:back)
AOS*x*flipXcoord	If *true* X coordinate is flipped
AOS*x*flaoSyncTimeout	Timeout for synchronization with FLAO arbitrator
AOS*x*argosRotIndex	Index of rotator for ARGOS (0:front, 1:center, 2:back)
AOS*x*argosObsEP	ICE endpoint for ARGOS Observation Arbitrator
AOS*x*argosOperEP	ICE endpoint for ARGOS Operation Arbitrator
AOS*x*argosSyncTimeout	Timeout for synchronization with ARGOS arbitrator

13 AOS GUI

Although AOS is essentially a thin interface layer to allow TCS subsystems to operate the AO system, and thus it should be totally controlled by the instrument software through the IIF, it is anyway provided with a GUI which is intended to be up and running at the operator console during observations. Its design is based on a set of requirements provided by potential users [6]. A previous document describing the AOS GUI [7] is now obsolete.



FLAO: Connected
ARGOS: Disconnected

Power: Off
TSS: Disabled
Coils: Disabled

Offload: 0.00 0.00 0.00
Modes: 0
Skip frame: 0 %

```
Thu Sep 29 12:42:19.002 2016 LSS is not running
```

Figure 3: AOS GUI Main Panel

The main task of the AOS GUI is informative: it provides access to a set of parameters which allow the operator to verify the good health of the AO system and the correctness of the operations.

Other than that, a small number of buttons allows the TO to perform the main housekeeping operation on the AO System: switch on/off the WFS, switch on/off the Adaptive Secondary and so on.

The AOS GUI includes also a command panel (usually hidden) from which commands usually received from TCS subsystems can be manually sent to the FLAOSup.

13.1 Main panel

Figure 3 shows a snapshot of the main panel of the AOS GUI which, except for a few buttons for general operations has essentially an informational purpose.

The AOSGUI main panel can be divided in a few section which are described in the following sections.

13.2 AOSGUI Top Section

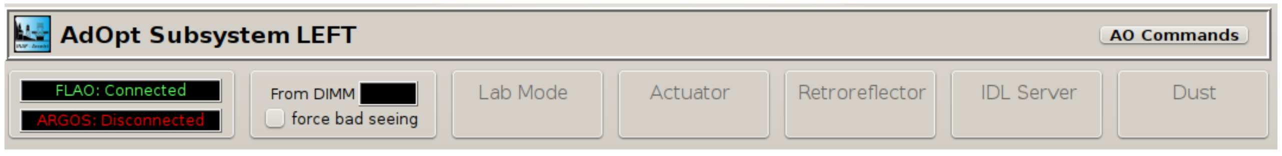


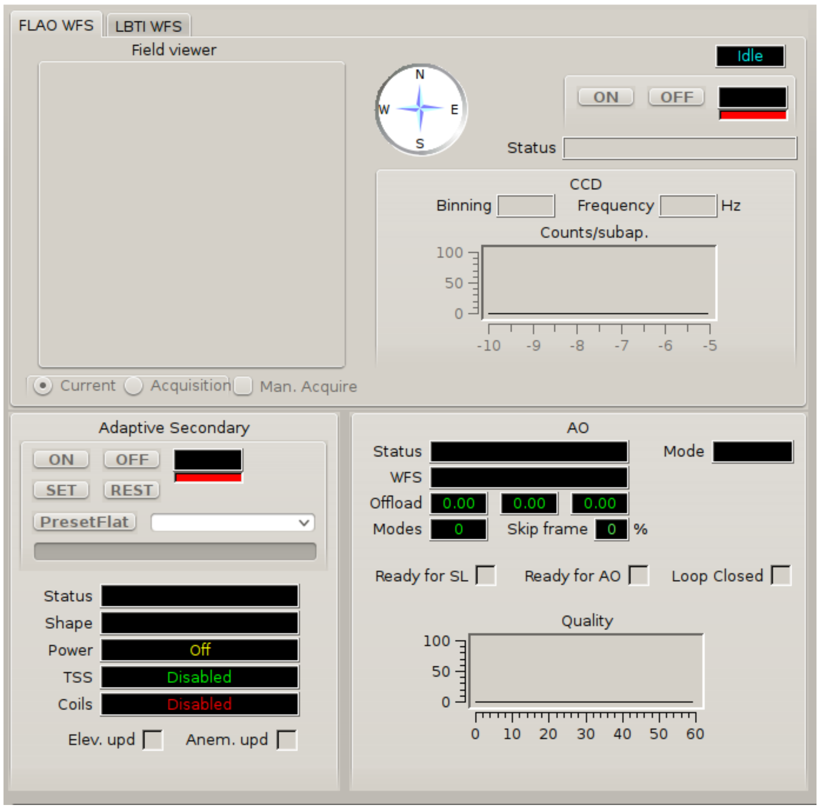
Figure 4: AOSGUI Top Section

The top section is shown in figure 4 and includes two horizontal subsections.

At the top of the section you may find a label line containing the identification of the panel (name and side), and a button (**AO Commands**) to be used to open the Command Panel described in Section 13.6.

Below you see a line with various status items. From left to right:

- **Communication Status**: shows the status of communication with the two external subsystems (FLAO and ARGOS).
- **DIMM**: indicating the seeing value from the DIMM; and the **Bad seeing** selector (see: Section 5.1.2).
- **LabMode**: when yellow indicates that the FLAO is operating in ”Lab Mode”. This variable is set by FLAOSup when it is operating in Lab mode: i.e.: several security interlocks are disabled to allow to perform some maintenance operations of the FLAO subsystem.
- **Actuator**: Indicates a failed actuator.
- **Retroreflector**: when yellow indicates that the FLAO subsystem is operating in ”retroreflector mode” (see: Section 11).
- **IDL Server**: when red indicates a failure of the IDL server used by the FLAO subsystem.
- **Dust Contamination**: when yellow indicates that the FLAO subsystem has detected a dust contamination problem on the Adaptive Secondary.



Status
Shape
Power Off
TSS Disabled
Coils Disabled

Status
WFS
Offload 0.00 0.00 0.00
Modes 0 Skip frame 0 %

Figure 5: AOSGUI FLAO Section

13.3 FLAO Sub-panel

At center left of the AOS GUI you may find the FLAO sub-panel shown by itself in figure 5. The panel contains the following areas:

1. **Wavefront sensor section**, displaying data related to the Wavefront sensor. It is provided with two panels, one for the FLAO WFS and the other for the LBTI WFS. Each panel includes a snapshot of the technical viewer[33], a section for the control of the WFS hardware with a few elements showing the current status, and a CCD subsection showing parameters related to the WFS-CCD.

2. **Adaptive secondary section**, containing a section for the control of the AdSec hardware, and a section with a number of status parameters.

3. **AO system section**, to the right of the AdSec section, displaying data and parameters related to the whole Adaptive Optics System. It includes an operating mode indicator, the number of corrected modes, three values showing the amount of "offload" currently required[34]. It also include space for a temporal graph of some image quality indicator[35]. Three more on/off indicators

[33]Due to operating requirements of the WFS subsystem, the technical viewer receives enough light to generate a useful image only before the AO loop is closed. Thus, the TV image is available only when the AO loop is open.
[34]The two tip-tilt components and the coma.
[35]Currently not implemented.

provide some global status information.

13.4 ARGOS Section

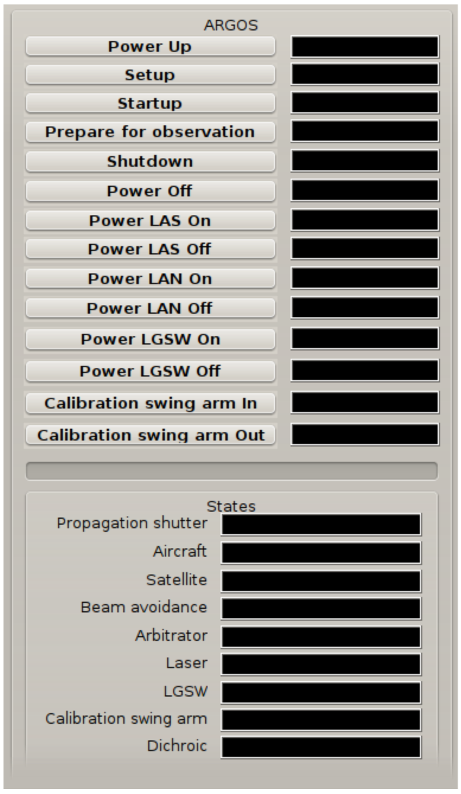


Figure 6: AOSGUI ARGOS Section

At center right of the AOS GUI you may find the ARGOS sub-panel (see: Figure 6).

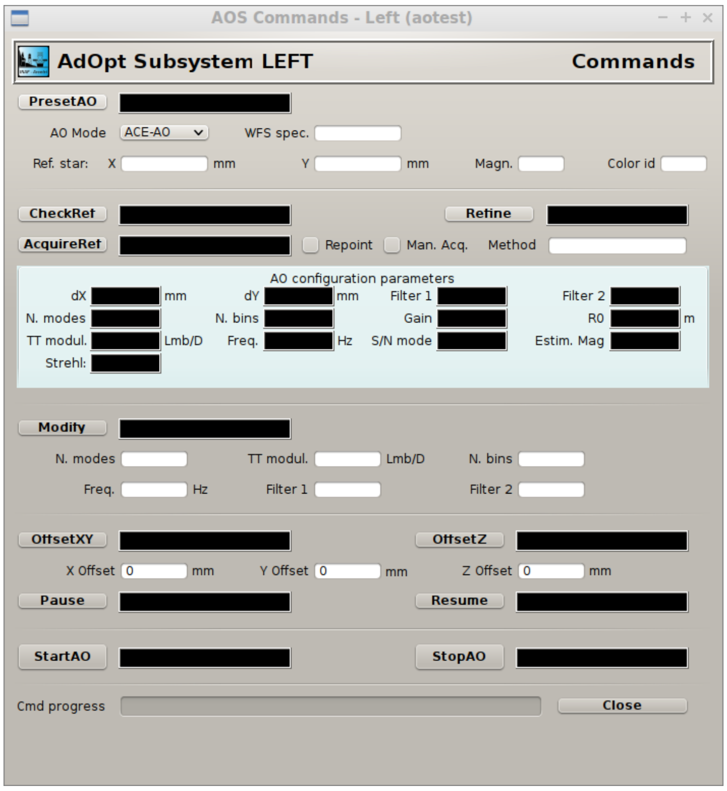
The section contains all buttons needed to operate the ARGOS subsystem and, in the bottom part, a set of status variables.

13.5 AOSGUI Log Section

The bottom part of the main AOS GUI panel (see: Figure 3) is used to show relevant log messages extracted from LSS.

13.6 Command panel

The Command panel of AOS GUI can be activated by the button quoted above and is shown in figure 7.



AO Mode: ACE-AO

X Offset: 0

Y Offset: 0

Z Offset: 0

Figure 7: AOS GUI command panel

The purpose of the command panel is twofold.

It provides the capability to test the AOS and the interactions with the FLAOSup independently from the rest of TCS and, which is more interesting during the operative life of the system, allows an operator a manual intervention when some procedure fails for unexpected reasons.

The command panel provides a set of buttons which correspond to most of the operating commands defined for the AOS described in Sections 5.2.1 through 5.2.13.

A few of the commands have associated arguments which may be modified before sending the command. Because the panel is intended either for debugging or for unexpected conditions, there are hardly preliminary checks applied to commands issued from the panel or to values sent as arguments. All the security checks are performed by the FLAOSup and any illegal or potentially dangerous command is simply rejected with an error message.

A Source Files Identification

The main AOS related source files are listed and briefly described in table 23.

Table 23: AOS Source Files

`AOS.cpp` `AOS.hpp`	Implementation of the AOS subsystem framework. It essentially provides to initialize the processes and the main control loop. The actual functionality are implemented in `ARGOSifApp` and `FLAOifApp`.
`AOSClear.cpp` `AOSClear.hpp`	Contains various function to clear or preset the Data Dictionary variables.
`AOSConfig.cpp`	Contains the code for methods to get data from configuration files.
`AOSEvent.cpp` `AOSEvent.hpp`	Contains classes derived from the TCS Event class to manage all events used in AOS.
`AOSInfo.cpp` `AOSInfo.hpp`	Implements the TCS information functions quoted in section 7.2.
`AOSTelemetry.cpp` `AOSTelemetry.hpp`	Contains the code to manage the telemetry recording.
`ARGOSifApp.cpp` `ARGOSifApp.hpp`	Implementation of the interface for the ARGOS external subsystem.
`FLAOCommand.cpp` `FLAOCommand.hpp`	Part of the implementation of the interface to the FLAO external subsystem. The main part is in file `FLAOifApp.cpp` and `FLAOifApp.hpp`.
`FLAOifApp.cpp` `FLAOifApp.hpp`	Implementation of the interface for the FLAO external subsystem. Main section, part of the implementation is in files `FLAOCommand.cpp` and `FLAOCommand.hpp`.
`GENifApp.cpp` `GENifApp.hpp`	Utility class used by both `ARGOSifApp` and `FLAOifApp`.
`Main.cpp`	Just launches the subsystem task and cleans up at the end.
`VarUtils.cpp` `VarUtils.hpp`	Contain code for the implementation of variable mirroring back and forth.
`Version.hpp`	Defines the version number and maintains documentation of the history of modifications.

The AOS code includes several more files distributed in a few subdirectories. Table 24 describes the content of each relevant subdirectory.

Table 24: Relevant subdirectories content

`aosgui`	All source files required to build the AOS GUI
`aosupervisor`	All source files related with interfacing with the FLAO external subsystem, plus the code required for FLAOSup emulation to be used for tests.
`ice`	ARGOS ICE interface definitions.
`test`	Test programs and procedures (see: Section B).

B Emulators and Test Procedures

The source code tree of AOS includes some programs and procedures which are not intended to be used during TCS operation, but can be built and used for tests.

B.1 AOS Test Client

An AOS test client (file: `.../aos/text/aosclient.cpp`) can be manually run to send commands to AOS from the command line emulating a TCS subsystem.

It is usually executed as:

```
./aosclient r/l
```

B.2 FLAO Emulator

The directory: `../aos/aosupervisor/emulator`) contains several source files used to build a bare bone emulator of FLAOSup[36]

The building and usage of the emulator is described in a `README` file included in the set.

B.3 ARGOS Emulator

Two python files in directory `.../aos/test`: `arbitrator_dummy.py` and `arbitrator_gui.py` can be used to emulate the ARGOS arbitrator.

The emulator can be launched from the same directory as:

```
python arbitrator_gui.py
```

[36]The source files and procedures provide support both for the older version of FLAOSup (pre UAO) and the current one, but the latter is very preliminary and not yet really usable.

The corresponding end points in AOS configuration file (see: Section 12.1) are:

```
DUMMY_ArgosObservationArbitrator -e 1.0:tcp -h 127.0.0.1 -p 15000
DUMMY_ArgosOperatorArbitrator -e 1.0:tcp -h 127.0.0.1 -p 15000
```

The emulator GUI allows the sending of several ARGOS external commands to AOS and shows commands received from AOS.

References

[1] Luca Fini, Lorenzo Busoni, and Alfio Puglisi. AOS Functional Description. Technical Report 481f300, INAF-Arcetri, May 2009.

[2] Roberto Biasi, Mario Andrighettoni, and Daniele Veronese. LBT672 Design Report: Electronics. Technical Report 641a006, Microgate, May 2008.

[3] Luca Fini, Alfio Puglisi, and Lorenzo Busoni. Integration of the AdOpt Software into TCS. Technical Report 486f004, INAF-Arcetri, Nov 2005.

[4] Lorenzo Busoni, Simone Esposito, Luca Fini, Alfio Puglisi, Armando Riccardi, and Marco Xompero. AO Supervisor - Functional Description. Technical Report 486f009, INAF-Arcetri, Mar 2009.

[5] Luca Fini. MsgD-RTDB. A middleware process for the AO Supervisor. Technical Report AdOptSW.012, in preparation, 2012.

[6] Douglas Miller. Adaptive Optics Subsystem (AOS) GUI Requirements. Technical Report 481s302, LBTO, Dec 2009.

[7] Luca Fini and Francesco Tribioli. AOS GUI Description. Technical Report AdOptSW.010, INAF-Arcetri, Dec 2007.

```
Doc_info_start
Title:
Document Type: Specification
Source: Osservatorio di Arcetri
Issued by: Luca Fini
Date_of_Issue: 29 Oct. 2016
Revised by:
Date_of_Revision:
Checked by:
Date_of_Check:
Accepted by:
Date_of_Acceptance:
Released by:
Date_of_Release:
File Type: PDF
Local Name:
Category: 400
Sub-Category: 480
Assembly: 481 Telescope Control Software
Sub-Assembly:
Part Name:
CAN Designation: 481f301
Revision: e
Doc_info_end
```