LBT-ADOPT
TECHNICAL REPORT

Doc.No : 481f301e
Version : 1.3
Date : ?? ??? 201?

AOS
The Complete Guide

Luca Fini, Alfio Puglisi, Lorenzo Busoni

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 functionalities, 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).

Contents

Glossary of terms and acronyms

AdSec. Short for Adaptive Secondary Mirror. In this context usually refers to the group of *AO 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 *AO Supervisor* components which control the Adaptive Secondary hardware.

AdSec Arbitrator. The Adaptive Secondary Arbitrator. A component of the *AO 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.

AO 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 some auxiliary devices (such as networking hardware) and includes the *AO Supervisor* and the real-time software.

AO Arbitrator. A component of the *AO 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*.

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

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

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

AOArb. Short for AO Arbitrator.

AOSup Short for AO 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 *AO 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. AO Real Time Database, the *AO 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 *WFS* to measure the light wavefront deformation (also: *CCD 39*).

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

WFS Arbitrator A component of the *AO 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.

Foreword

This document is divided into two parts. Part I describes AOS architecture and code structure, and includes details about the interaction between AOS and TCS, on one side and the AO Supervisor, on the other. Part II, describes the operational functions implemented by AOS to support the Adaptive Optics System. This second part is directly derived from a previous document (CAN 481f300 [1]) which is now obsolete.

This description of the AOS is related to AOS Version 10.x, corresponding to the version released at the end of the commissioning of the FLAO system #2.

Part I

AOS/AO Supervisor Software Architecture

1 Introduction

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 AO Supervisor[1]. It provides all the functionalities 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 AOSup and the TCS as a whole. It defines a set of commands which can be used by other TCS subsystems[2] to operate the AO System during an observation and supports the continuous mirroring of a small set of AOS related variables from the TCS Data Dictionary to the AOSup RTDB, and vice-versa. Although the main flux of control is originated from the TCS and the AOSup operates mainly as a slave, AOS also allows the AOSup to issue a few commands[3] to the TCS.

1.2 The AO Supervisor

1.2.1 AO Supervisor Architecture

A detailed description of the software architecture of the AO 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 1 shows the overall architecture of the AOSup, and its relationships with the TCS.

The AOSup is based on a distributed architecture where several independent processes cooperate in order to properly manage the underlying hardware, while ensuring safe operations of any device. The organization of processes in the AOSup reflects the splitting of FLAO system in two subsystems: the Adaptive Secondary (AdSec) and the Wavefront Sensor (WFS). The two software subsystems are currently hosted by two independent servers[4] indicated in the figure by the two large boxes named, respectively, `adsecdx` and `wfsdx`[5]. Also shown in the figure is a second server named `lbti-dswfs`

[1] Please note that in the following pages we will use the term **AOS** to indicate the Adaptive Optics Subsystem, i.e.: the software component of the TCS which communicates which the AO Supervisor, while we will use the term **AO System** to indicate the Adaptive Optics System as a whole. The term **AO Supervisor** or AOSup, indicates the software controlling the entire AO System.

[2] The TCS subsystems currently using the AOS commands are the IIF and the AOS GUI.

[3] This feature is used essentially to support the "Offload modes" function and can be used to allow the control of the AdSec hexapod from AO Supervisor procedures. See sections 9 and 10.1.

[4] 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 AOSup could run on a single server.

[5] The names refer to the right side AO System. For the left side the AO Servers are named `adsecsx`, `wfssx`, and so on.

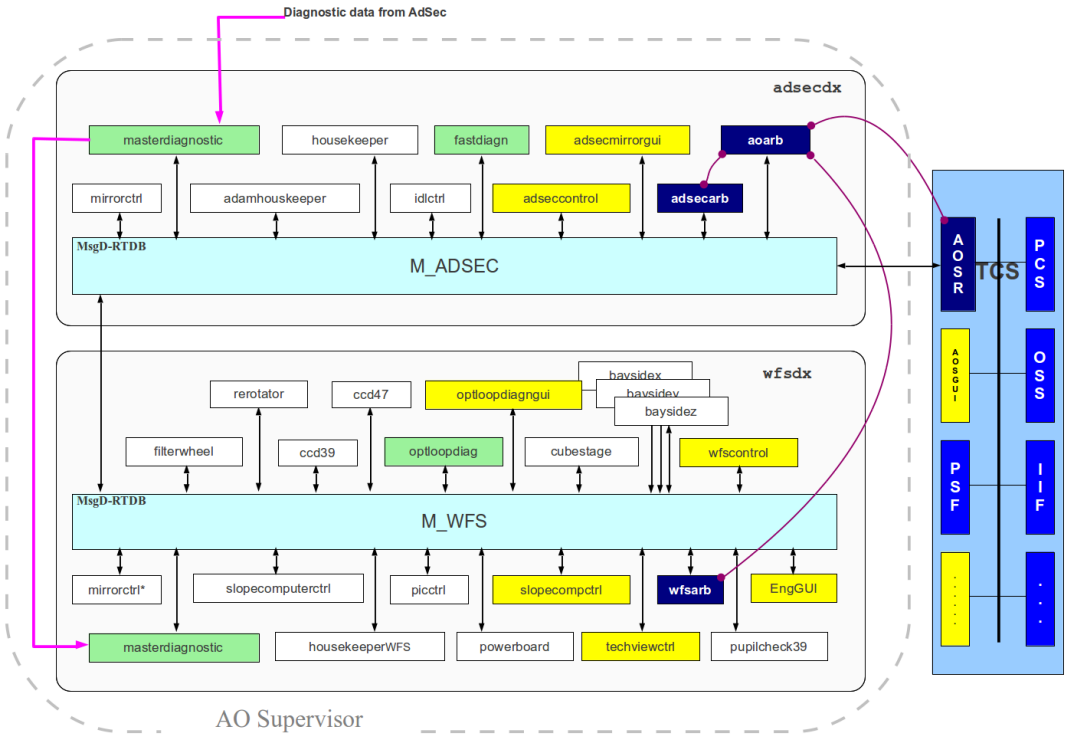

Figure 1: AO Supervisor architecture and interactions with TCS

which runs the processes related to the management of the LBTI WFS. This is in any respect a clone of the FLAO WFS subsystem.

All interactions between AOSup processes are supported by a central process named MsgD-RTDB[6] 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 MsgD-RTDB processes can communicate among them 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 AOSup 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 1 by the violet lines.

1.2.2 AO System operating modes

When supporting an observation, the AO System has four different operating modes:

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

- **FIX-AO**, when operating in "seeing limited" mode. In other words, this is not an "adaptive" mode in that the shape of the adaptive mirror is fixed, but is anyway supported by the AO System. When in FIX-AO mode the AOSup can operate without the support of the WFS subsystem.
- **TTM-AO**, when operating in adaptive mode, but correcting only the tip-tilt components of the atmospheric aberration.
- **ACE-AO**, when operating in full adaptive mode with an automatic selection of AO loop parameters.
- **ICE-AO**, when operating in full adaptive mode and providing means to allow the observer to adjust AO loop parameters[7].

The AOSup can be operated according to the above operating modes by means of a set of commands issued by the AOS and received and executed by the AO Arbitrator.

1.2.3 AO Arbitrator Commands

The set of commands which AOS can use to control the AO System is shown in table 1. By using these commands properly (i.e.: in the proper sequence) the AOS can control all the operations that are needed to successfully perform an observation.

Note that AOS defines its own set of commands which are provided to other TCS subsystems to operate the AO System, but the AOS command set is not exactly matching the AO Arbitrator set: in order to adapt the internal logic of the AO System to the operational scheme of TCS some AOS commands are implemented as short procedures, mixing together commands issued to the AO Arbitrator and internal TCS commands. More details related to AOS commands and their implementation can be found in section 8 and, extensively, in the second part of this report.

1.2.4 AO Arbitrator Command Sequences

In order to properly operate the AO System, a specific sequence of commands must be issued from the AOS to the AOSup. The AO Arbitrator is driven by a Finite State Machine (whose diagram is shown in figure 2) which ensure that commands are provided in legal sequences.

The main sequence of commands to perform an adaptive mode observation, starting from *Operational* status, consists of the following steps:

1. `PresetAO`: provides the AOSup with values for all parameters which are needed to prepare for AO; moved to *ReadyToAcquire* status.
2. `AcquireRefAO`: starts the acquisition of the reference star. When finished moves to *RefAcquired* status.
3. `StartAO`: closes the AO loop.

[7] ICE-AO operating mode has not yet been fully implemented.

Table 1: Commands accepted by the AO Arbitrator

Command	Description
`AcquireRefAO`	Acquire the reference star and become ready for closing the AO loop
`AdjustGain`	Change the loop gain
`CheckRefAO`	Returns offset values between reference star actual and nominal position
`CorrectModes`	Apply mirror shape correction while in closed loop
`ModifyAO`	Modify some AO loop parameter
`PresetAO`	Preset AO System for adaptive operation
`PresetFlat`	Preset AO System for seeing limited operation. The AO System at startup has already a default seeing limited shape. This command may be used to select different shapes, e.g.: optimized for specific instruments and/or operations.
`OffsetXY`	Offset AO pointing[a]
`OffsetZ`	Offset AO focus[a]
`PauseAO`	Temporarily suspend the adaptive correction loop
`RefineAO`	Perform optimization of AO loop parameters
`ResumeAO`	Resume a suspended adaptive correction loop
`SetZernikes`	Apply mirror shape correction while in seeing limited mode
`StartAO`	Start the AO mode (i.e.: close the AO loop)
`Stop`	Stop current operation

[a]When this command is issued in closed loop, the offset value is limited to about half an arcsec, due to the limited speed of the offload modes mechanism.

As shown in figure 2 in each operational status the AO Arbitrator may accept some commands other than the one defined in the main sequence. As an example, the `OffsetXY` command is legal both in *LoopClosed* and in *LoopSuspended* status. This is needed to support target offsetting of any length: when the desired offset is less than one half of an arcsec, the offset command can be issued maintaining the loop closed. For larger offsets, instead, the following sequence of commands must be issued:

1. `Pause`
2. `OffsetXY`, while an opposite offset is requested to the telescope pointing system
3. `Resume`

The task of implementing this sequence is to the subsystem controlling the AOS (typically the IIF).

1.2.5 Safety issues

During the operation of the Adaptive Mirror a number of safety issues must be taken into account, among those a few require a special support from the AOS[8]:

[8]AOS versions prior of 9.3 relied on TCS support also for getting wind speed data from the AdSec dedicated anemometer. Currently the anemomenter is integrated in the AO System and data acquisition is performed by a specific AOSup process.

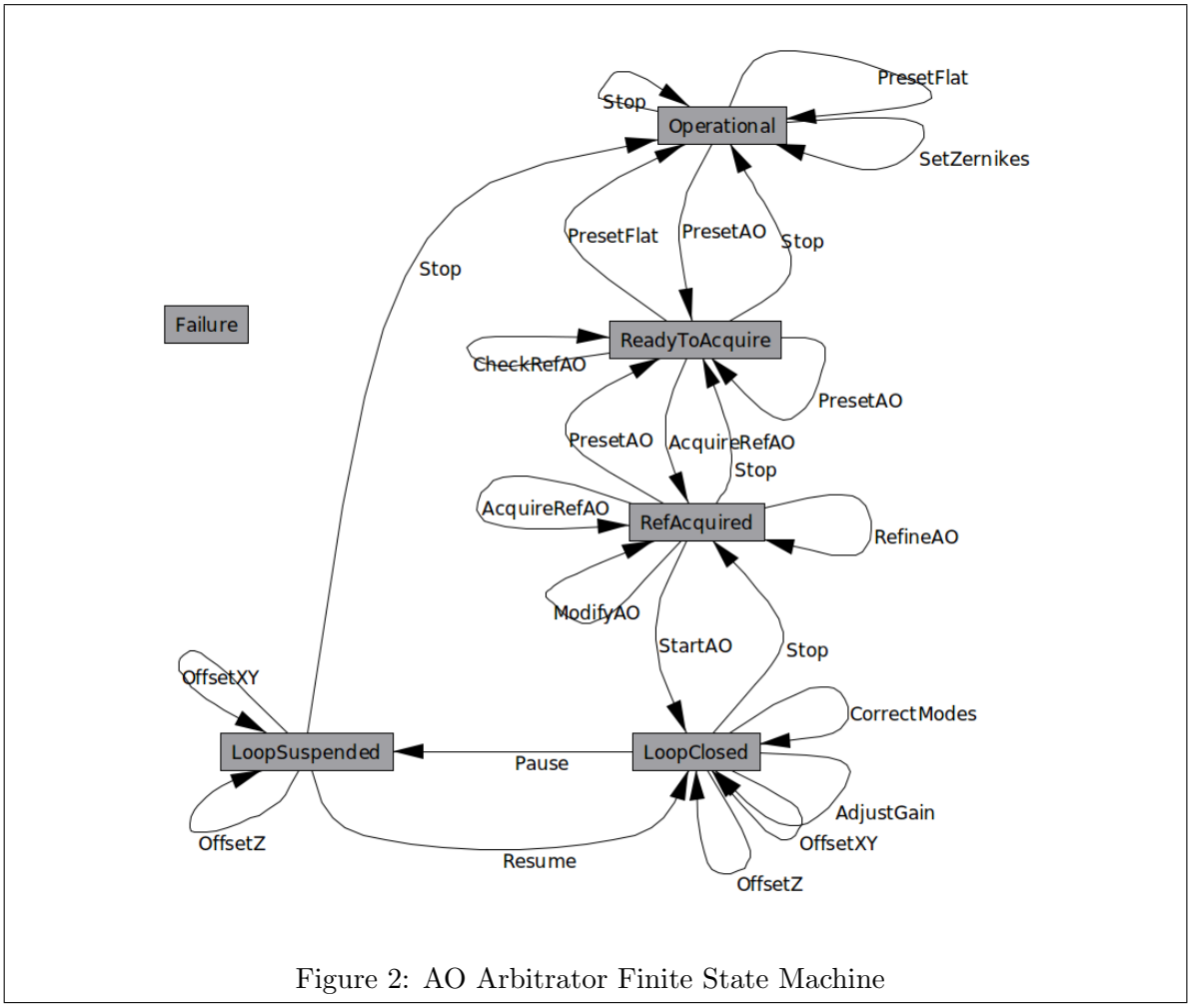

Figure 2: AO Arbitrator Finite State Machine

1. **Telescope elevation**: the mirror must be set in safe position if the telescope moves below 20 degrees of elevation.
2. **Swing arm position**: the mirror must not be operated when the swing arm holding it is not in the optical path position.

The above conditions must be securely detected to take the proper actions. Note that while the windspeed is measured by a specific device placed in the vicinity of the secondary mirror and the related data are provided by some AOSup module, for telescope elevation and swing arm position, AOS provides the required support by maintaining related variables in the RTDB and implementing an expiration mechanism for those variables (see section 4). This implies that the AOSup cannot be operated if the AOS (and at least a subset of other TCS subsystems) is not up and running[9].

[9]This safety feature can be overridden by enabling "lab mode", see also section: 4.2.2.

2 AOS Architecture

The AOS is a standard subsystem of TCS. As such it is globally structured like any other TCS subsystems. As detailed in the following sections, AOS is divided essentially into three modules: `Main`, `AOSSubsystem`, and `AOSClient`.

2.1 Main

The Main module[10] performs the following steps:

1. Gets configuration data; e.g.: the IP number of the machine where the AOSup `MsgD` is running, the initial debug level, etc. (for a list of all configuration parameters see section 11).
2. Initializes the telemetry collection system.
3. Starts the `AOSSubsystem`'s thread.
4. Initializes the AOS commands.
5. Waits for `AOSSubsystem` to terminate.

2.2 AOSSubsystem

`AOSSubsystem`[11] 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, `execute()` is essentially a loop which controls the communication rendez-vous with the `MsgD`.

The loop performs five simple steps:

1. Checks for a terminate request
2. Instantiates an `AOSAoApp` object.
3. Starts `AOSAoApp` by executing its `Exec()` method.
4. Destroys the `AOSAoApp` object.
5. Sleep 1 second

The `AOSAoApp` object is derived from the `AOApp` class provided by the AOSup software. When started it tries to connect as a client to `MsgD` waiting indefinitely for a reply. When the communication is established, the `AOSAoApp` finishes up its initialization and begins its operations by calling the `Exec()` method.

[10] Main module code is in files `Main.cpp`, `Main.hpp`.
[11] AOSSubsystem code is in files: `AOS.cpp`, `AOS.hpp`.

The `Exec()` method returns only when the communication with `MsgD` terminates (e.g.: because the AOSup is stopped) or when an explicit stop is requested by the operator. In this case the loop described above is iterated and AOS returns to wait for communication with `MsgD`.

Because most of the functionality of the AOS is embedded into the class `AOSAoApp`, the latter is described in deeper details in section 3.

2.3 AOSClient

`AOSClient`[12] implements the class to be used by clients (on the TCS side) to send commands to the AOS. This follows closely the TCS standard for client communication and its purpose is simply to receive from other TCS subsystems commands to be translated into commands to be delivered to the AOSup (see sect. 8).

3 AOSAoApp

`AOSAoApp`[13] is the class at the heart of the AOS; it is derived from the standard `AOApp` class used to implement clients for the AOSup and provides all the functionalities needed for the communication between AOSup components. It implements a number of base functionalities such as message queue management, automatic replies to several housekeeping messages, error logging, and more.

Any application based on `AOApp` is intrinsically multi-threaded, although this is hidden into the implementation and multi-threading is usually not directly managed by the application programmer.

After being instantiated `AOSAoApp` executes its `Exec()` method which performs a standard initialization procedure as follows:

1. Installs handlers for asynchronous messages coming from the AOSup.
2. Waits for `MsgD` to accept a connection.
3. Executes the `Run()` method.

The Run() method is where the functionalities of `AOSAoApp` are implemented and is essentially a loop which terminates when the communication with `MsgD` is interrupted for any reason or when receiving from some Supervisor client an explicit termination command[14]. When the `Run()` method terminates, the `Exec()` method as a consequence is also terminated and the control returns to `AOSSubsystem`, as described in section 2.2.

While connected as an `MsgD` client the AOS registers itself either with the name `AOS.L` (left side) or `AOS.R` (right side).

[12] The source code is in files: `client/AOSClient.cpp` and `client/AOSClient.hpp`.

[13] The related source code is in files: `AOSAoApp.cpp`, `AOSAoApp.hpp`.

[14] This is a standard feature of all Supervisor clients and is used to properly shutdown all the client processes when the AOSup is terminated

3.1 Handlers

Handlers are special methods[15] of `AOSAoApp` which manage asynchronous messages coming from the AOSup. They manage messages related to the following functions:

- `arbalert_hndl`: Receives "alert" notifications from the AO Arbitrator
- `varnotify_hndl`: Receives notifications of relevant RTDB variables change.
- `hexapod_hndl`: Receives and executes hexapod related commands.
- `housekeep_hndl`: Receives and executes housekeeping commands used for debugging purposes.
- `offload_hndl`: Receives and reissues "offload" requests (see section 9).
- `default_hndl`: Receives all other messages directed towards the AOS.

Handlers are installed as part of `AOSAoApp` 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.

3.1.1 `arbalert_hndl`

Alerts are messages sent by arbitrators to notify error conditions which are not the consequence of some error in command execution, but are dependent on external conditions such as hardware failures and the like.

The purpose of the corresponding handler is currently to decode the error message and properly display it on the operator GUI and log it in the TCS log system. If the need arises alert messages could also be used to perform some specific operation, e.g.: to force a stop of the current operation.

3.1.2 `varnotify_hndl`

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 `AOSAoApp` registers with the `MsgD` to be notified of updates of a specified set of variables.

Whenever any AOSup 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 4.

[15] The related source code is in files: `AOSHandlers.cpp` and `AOSHandlers.hpp`.

3.1.3 hexapod_hndl

This handler manages hexapod related commands[16] sent from the AOSup.

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 **??**.

3.1.4 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 10.2.

3.1.5 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 9.

3.1.6 default_hndl

The default handler will receive all messages not trapped by other handlers. It is curently used to manage the `TERMINATE` command[17].

All other commands possibly received by the default handler are unexpected and thus will only generate warnings.

3.2 AOSAoApp::Run()

After a successful instantiation and `AOSAoApp` and its registration as a client of `MsgD`, the `Run()` method is executed.

The `Run()` method is the main loop of the AOS, despite the fact that most of the work is performed elsewhere, i.e.: by handlers for commands originating from the AOSup, and by the TCS Command Interface code for commands originating from other TCS subsystems.

When started the `Run()` method performs the last steps of `AOSAoApp` initialization and enters the main loop where, at a convenient rate[18], the following tasks are performed:

[16] 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.

[17] All AO Supervisor clients are required to properly manage the `TERMINATE` command. It is not intended for normal use but only in debugging or troubleshooting operations.

[18] Currently the main loop rate is about 1 Hz.

1. Checks the current status of the AOSup and sets the AOS status accordingly. This is done by the `synchronize()` method (see below).
2. 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`. This is done by the `updateTCSVariables()` method.
3. Checks the status of the WFS control processes[19] and updates the corresponding status variables accordingly. This is done by the `checkWFS()` method. Whenever any of the WFS subsystems comes into life a new request for notification of related variables is issued.

In order to derive the current status of communication, the `synchronize()` method makes a number of tests: communication with the `MsgD`, presence of the `AOARB` arbitrator, value of the "Connection status" variable. Possible results of the checks are the following:

- NO CONNECTION: no communication with `MsgD` has been established.
- NO ARBITRATOR: communication with `MsgD` has been established, but the AO Arbitrator is not running.
- STANDALONE: both `MsgD` and the AO Arbitrator are up and running, but the AO Subsystems will not accept commands from AOS.
- OPERATING: the AOSup is up and running and willing to receive commands.
- SIMULATION: the AOS was started in simulation mode, no connection to `MsgD` is attempted.

As a by-product of the above checks, if the `MsgD` fails to respond within short time, the communication cycle of the `AOSAoApp` is terminated as detailed in section 3.

4 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[20] variables are grouped accordingly to their functions.

4.1 Notes regarding variable tables:

1. The tables include the name of each variable either in the TCS `DD` (above) or in the `RTDB` (below), or both when there is correspondance.
2. For each variable it is indicated the source of the datum (TCS if it is from the `DD` or SUP if it is the AO Supervisor), the type and the size if greather than one, and a brief description.
3. Variable types are indicated as `str`, `int`, `real` corresponding in `DD` to,respectively, `string`, `long`, `double`.

[19] Currently two WFSs are supported FLAO WFS and LBTI WFS.
[20] A few variables, used exclusively for internal uses of the AOS code, are omitted.

Table 2: AOS housekeeping variables

aos.side[s].connected	TCS	bit	Indicates when the AOS is connected to MsgD
aos.side[s].conntime	TCS	str	Time of last connection to MsgD
aos.side[s].idlstat AOARB.x.IDL_STAT	SUP	str	Flag indicating that the IDL subsystem is running
aos.side[s].labmode AOARB.x.LAB_MODE	SUP	long	Flag indicating when the AOSup is in "Lab Mode" (see sect. 1.2.5)
aos.side[s].man_acquire	TCS	int	Set to 1 by AOSGUI to request manual selection of reference star (see sect. 4.2.4)
aos.side[s].msgdident	SUP	str	MsgD identification string
aos.side[s].msgdip	TCS	str	IP address of machine running the MsgD
aos.side[s].msgdstat	TCS	str	MsgD current status (the result of the syncronize() internal loop)
aos.side[s].ref_xy x.REF_XY.wfs	TCS	int[2]	Holds the last value of reference star position (see sect. 4.2.4)
aos.side[s].rotIndex	TCS	int	Rotator identifier (from configuration file; 0: front, 1: middle, 2: rear)
aos.side[s].rr_enabled AOARB.x.RR_ENABLED	SUP	int	Set to 1 when the retroreflector is installed (see sect. 4.2.1)
aos.side[s].running	TCS	bit	Indicates when the AOS is connected to MsgD and communicating with the AOArb
aos.side[s].servstat AOARB.x.ServStat	SUP	str	Current AOSup communication status
aos.side[s].sstat_color	TCS	int	Server status color index (used only by AOS GUI)
aos.side[s].starttime	TCS	str	Time of AOS start
aos.side[s].tel_enabled	TCS	bit	Indicates when the telemetry collection is enabled

4. Values generated by the AOSup and mirrored to the DD are updated via the notification mechanism, as explained in section 3.1.2. Values originating in the TCS and mirrored to the RTDB are updated by the polling mechanism described in section 3.2.

5. A few values communicated to the RTDB are functions of more than one DD variable (e.g.: ISTRACKING is a boolean value derived from the combination of three status variables: mcs.azDrive.trackingMode, mcs.elDrive.trackingMode and mcs.onSource). All these are listed in table 4.

Table 3: Miscellaneous TCS variables mirrored to MsgD

env.weather.lbt.windDirection AOS.EXTERN.WINDDIRECTION	TCS	real	External wind direction (external anemometer)
env.weather.lbt.windSpeed AOS.EXTERN.WINDSPEED	TCS	real	Wind speed (external anemometer)
gcs.side[s].GuideCam.centroid_FWHM_X AOS.x.GUIDECAM.CENTROID.X	TCS	real	Guide camera centroid X
gcs.side[s].GuideCam.centroid_FWHM_Y AOS.x.GUIDECAM.CENTROID.Y	TCS	real	Guide camera centroid Y
iif.DIMM.seeing AOS.DIMM.SEEING	TCS	real	Seeing value from the DIMM
mcs.azDrive.position AOS.TEL.AZ	TCS (**T**)	real	Current telescope azimuth
mcs.elDrive.position AOS.TEL.EL	TCS (**T**)	real	Current telescope elevation
mcs...rotators[n].actualPositionASec AOS.x.ROTATOR.ANGLE	TCS	real	Rotator angle for currently selected rotator
oss.side[s].adsc.abs_pos AOS.x.HEXAPOD.ABS_POS	TCS	real[6]	Hexapod absolute position
oss.side[s].terc.abs_pos AOS.x.TERTIARY.ABS_POS	TCS	real[4]	Tertiary mirror absolute position
pcs...achieved.achieved_DEC.Radians AOS.TEL.DEC	TCS	real	Current telescope declination
pcs...achieved.achieved_RA.Radians AOS.TEL.RA	TCS	real	Current telescope right ascension

6. 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 AOSup to take actions based on variable expiration[21].

7. In variable names related to TCS DD the subscript character "s" is used to indicate the telescope side (current corresponding values are 0 for left side and 1 for right side).

8. In variable names related to AOSup RTDB the character "x" in the name also indicates the side, where needed. Its value may be either L or R for left and right side, respectively.

9. In variable names related to Wavefront Sensors, the suffix @wfs is used to indicate that the variable is to be found in the related MsgD, actual suffix can be either @FLAOWFS or @LBTIWFS; the corresponding DD variable will have a field named wfs*n* to indicate either flaow (FLAOWFS) or lbtiw (LBTIWFS).

10. Some names of variables in DD have been shortened using ellipsis (...) for typographical reasons.

[21] This mechanism is used, e.g.: for safety related variables as described in section 1.2.5.

Table 4: Functions depending on some TCS variables and mirrored to `MsgD`

`hbs_on()` `AOS.TEL.HBS_ON`	TCS	func	Set to 1 when HBS is on
`hexapod_status()` `AOS.x.HEXAPOD.STATUS`	TCS	func	Hexapod status.
`tel_guiding()` `AOS.TEL.ISGUIDING`	TCS	func	Set to 1 when telescope is guiding
`tel_tracking()` `AOS.TEL.ISTRACKING`	TCS	func	Set to 1 when telescope is tracking
`swa_deployed()` `AOS.x.SWA.DEPLOYED`	TCS (**T**)	func	Set to 1 when the swing arm is either completely deployed or completely retracted
`vent_on()` `AOS.x.TEL.VENT_ON`	TCS (**T**)	func	Set to 1 when the ventilation system is running

11. The source code relevant for the management of variables can be found in files: `VarUtils.cpp` and `VarUtils.hpp`.

4.2 Variable affecting AOS behaviour

Some variables are used to modify the behaviour of AOS, on request both of the observer (e.g.: from the AOSGUI) or of the AO Supervisor[22].

In the following sections the meaning of each variable is described in details.

4.2.1 `rr_enable`: retroreflector enabled

This variable is mirrored from a corresponding variable in `RTDB`. It is to be set from the AO Supervisor side. When set it affects the offload mechanism (see section 9) so that only tip-tilt components of the offload command are actually applied.

This is necessary to avoid corrections applied to the primary mirror which is actually outside the optical path.

4.2.2 `labmode`: lab mode enabled

This variable is mirrored from a corresponding variable in `RTDB`. It is to be set from the AO Supervisor side and results in ignoring the safety related `DD` variables. This allow the AO system to be operated even when the TCS is not providing elevation and swing arm status. Operating the AO System in this condition is dangerous and must be done only by properly trained personnel for maintenance and other "closed dome" activities.

The only effect on the AOS side is to display an alarm on the AOSGUI.

[22] In this case the engineering GUI's provide the way to select the proper behaviour.

Table 5: AO System related variables

aos.side[s].ao.ao_ready AOARB.x.AO_READY	SUP	int	Set to 1 when AO System is "ready for AO"
aos.side[s].ao.correctedmodes AOARB.x.CORRECTEDMODES	SUP	int	Number of corrected modes
aos.side[s].ao.loopon AOARB.x.LOOPON	SUP	int	Set to 1 when the adaptive loop is closed
aos.side[s].ao.mode AOARB.x.MODE	SUP	str	Current observation mode (FIX_AO, TT_AO, ACE_AO, ICE_AO.)
aos.side[s].ao.msg AOARB.x.MSG	SUP	str	Generic message string. To be displayed on AOS GUI
aos.side[s].ao.ofl_enabled AOARB.x.OFL_ENABLED	SUP	long	Set to 1 when the offload mechanism is enabled (see sect. 9
aos.side[s].ao.sl_ready AOARB.x.SL_READY	SUP	int	Set to 1 when AO System is "ready for Seeing Lmtd. operation"
aos.side[s].ao.status AOARB.x.FSM_STATE	SUP	str	Adaptive optics system status
aos.side[s].ao.strehl AOARB.x.STREHL	SUP	real	Estimated Strehl ratio. Currently unreliable
aos.side[s].ao.wfs_source AOARB.x.WFS_SOURCE	SUP	str	Wavefront sensor currently in use. Either "FLAO", the WFS for first light AO, or "LBTI"

4.2.3 man_acquire: enable manual selection of reference star

This variable can be set/reset from the AOSGUI. When set it enables the interactive selection of the reference stari (see section 12).

The interactive selection of reference star can be useful when the automatic star selection usually performed by the AcquireRef command fails due to crowded field.

4.2.4 idlstat: signal bad state of IDL subsystem

This variable is mirrored from a corresponding variable in RTDB. It is set by the AO Supervisor when the IDL processor is not working properly (usually due to some license problem).

When set it causes a red banner to be displayed on over the AOSGUI.

5 Events and logging

AOSAoApp is provided with 5 levels of log messages[23], in decreasing order of verbosity: trace, debug,

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

Table 6: Adaptive Secondary related variables - part 1

aos.side[s].adsec.anem_speed ANEM.x.SPEED	SUP	int[12]	Wind speed components
aos.side[s].adsec.anem_time	TCS	int	Wind speed update time
aos.side[s].adsec.anem_upd adsecarb.x.ANEM_UPD	SUP	int	Flag indicating proper functioning of AO internal anemometer
aos.side[s].adsec.coil_status adam...x.COIL_STATUS	SUP	int	Flag indicaiting that AdSec coils are enabled
aos.side[s].adsec.contamination ADSEC.x.CONTAMINATION	SUP	int	Mirror dust contamination detected
aos.side[s].adsec.elev_upd adsecarb.x.ELEV_UPD	SUP	int	Flag indicat ing that the AO Supervisor is receiving elevation data
aos.side[s].adsec.fl_filename	SUP	str	List of avaiable flat specifications (filename)

`info`, `warning` and `error`.

The first two levels (most verbose) correspond only to lines logged via the *Syslog* mechanism and can be disabled either by a configuration parameter (see section 11) or via run-time messages from the AOSup.

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.

6 Telemetry data

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

A list of data items stored into telemetry is shown in table 9.

7 TV image frames

The AOS GUI (see section 12) 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:

Table 7: Adaptive Secondary related variables - part 2

aos.side[s].adsec.health AOARB.x.ADSEC.HEALTH	SUP	int	Indicates that all processes relevant for the AdSec are up and running
aos.side[s].adsec.led adsecarb.x.LED	SUP	int	Indicates some status of the thin shell (0=off/1=safe/2=set)
aos.side[s].adsec.msg adsecarb.x.MSG	SUP	str	Generic message related to AdSec. To be displayed on AOS GUI
aos.side[s].adsec.nwact ADSEC.x.NWACT	SUP	int	Indicates the number of actuators which are not working
aos.side[s].adsec.offload	SUP	int[22]	Offload vector
aos.side[s].adsec.popmsg ADSEC.x.POPMSG	SUP	str	Generic popup message
aos.side[s].adsec.main_pwr_status adam...x.ADSEC.MAIN_POWER_STATUS	SUP	int	Set to 1 when adsec main power is on
aos.side[s].adsec.safeskip_perc adsecarb.x.SAFESKIP_PERCENT	SUP	str	Percentage of skipped frames in optical loop
aos.side[s].adsec.shape ADSEC.x.SHAPE	SUP	str	Currently selected mirror shape
aos.side[s].adsec.status adsecarb.x.FSM_STATE	SUP	str	Status of the AdSec FSM
aos.side[s].adsec.tss_status adamhousekeeper.x.TSS_STATUS	SUP	int	Set to 1 when mirror TSS system is enabled
aos.side[s].adsec.zern_lut		int	Currently unused
aos.side[s].adsec.zern_lut_time		int	Currently unused

```
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 AOSup. AOS GUI polls for changes in file access date and displays the file content whenever the date changes.

Table 8: Wavefront Sensor related variables. WFS1: FLAO

aos.side[s].wfsn.active	SUP	int	Set to 1 when the corresponding WFS is active (as a result of a SetNewInstrument command)
aos.side[s].wfsn.ccdbin ccd39.x.XBIN.CUR@wfs	SUP	int	WFS CCD binning
aos.side[s].wfsn.ccdfreq ccd39.x.FRMRT.CUR@wfs	SUP	real	WFS CCD current frequency value
aos.side[s].wfsn.counts optloopdiag.x.COUNTS@wfs	SUP	int	WFS CCD average counts per frame
aos.side[s].wfsn.enabled	TCS	int	Set to 1 when the corresponding WFS Arbitrator is communicating
aos.side[s].wfsn.filter1 filterwheel1.x.POSNAME.CUR@wfs	SUP	str	WFS filter 1 position
aos.side[s].wfsn.health AOARB.x.FLAOWFS.HEALTH	SUP	int	Flag indicating that processes relevant for the WFS are up and running
aos.side[s].wfsn.mod_ampl wfsarb.x.LED@wfs	SUP	int	Indicates global status of WFS. 0=off, 1=on
aos.side[s].wfsn.mod_ampl wfsarb.x.MOD_AMPL@M_WFS	SUP	real	WFS TT-mirror modulation amplitude
aos.side[s].wfsn.msg wfsarb.x.MSG@M_WFS	SUP	str	WFS related generic message
aos.side[s].wfsn.no_subaps slopecompctrl.x.NSUBAPS.CUR.@wfs	SUP	int	Number of subapertures in current WFS configuration
aos.side[s].wfsn.pyramid_pos wfsarb.x.PYRAMID_POS@wfs	SUP	int[2]	Position of WFS pyramid with respect to TV CCD
aos.side[s].wfsn.source	TCS	str	Current source of WFS data
aos.side[s].wfsn.tv_binning ccd47.x.XBIN.CUR@wfs	SUP	int	TV CCD current binning
aos.side[s].wfsn.tv_exptime ccd47.x.FRMRST.CUR@wfs	SUP	real	TV CCD current exposure time
aos.side[s].wfsn.tv_filter2 filterwheel2.x.POSNAME.CUR@wfs	SUP	str	TV CCD current fiter 2 selection

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

8 Commands

The AOS provides two command based interfaces; one is built over the AOSup command mechanism is used to receive commands from the AO Supervisor, the other is based on the standard TCS command mechanism and is used to provide AO services to the TCS.

Table 9: Data items store 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

8.1 Commands from the AO Supervisor

Commands received from the AOSup are used to support the "Offload modes" function (see section 9), to allow the AO Supervisor to directly operate the hexapod during some calibrations procedures (see section 10.1) or to perform housekeeping tasks not directly connected with the operation of the AO System (see section 10.2).

8.2 Commands from other TCS subsystems

AOS defines a set of commands to be used by other TCS subsystems to operate the AO System. These commands are described in full details in the second part of this report (see sections 13 through 15).

9 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.

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 a specific command sent by AOSup and received by the related handler (see page 13). The command argument consists of an array of 22 zernike terms which is managed by a specific method of the `AOSAoApp` object[24].

[24]The related code is in file `AOSAoApp.cpp` (method: `offload()`)

Each array received is managed as follows:

1. The command queue is checked to verify that no other commands are waiting. If more commands are waiting the current one is discarded[25] with a warning event.
2. The array is divided into two parts: the lower order part (elements 2 to 4: the first element is never considered) related to tip/tilt and focus terms, and the higher order part, including elements from 5 to 22. The two parts are independently compared against an array of threshold values.

   The `tip-tilt-focus` offload action is scheduled to be executed if the absolute value of any of the low order offload components is greater than the corresponding threshold. Analogously the `highorder` offload action is scheduled to be executed if the absolute value of any of the high order component is greater than the corresponding threshold.
3. If any offload action is to be executed, the offload vector is rescaled by multiplying it by a vector of gain coefficients[26] to adjust for differences in units used for zernike values in the AOS with respect to the AO Supervisor.
4. If a `tip-tilt-focus` offload action is scheduled, tip tilt and focus values are computed from the low order zernike terms and the correction is sent as a PSF command to be applied to the Hexapod.
5. If a `highorder` offload action is scheduled, the offload vector (with the first 4 components set to 0) is sent to the PSF to be applied to the primary support system.

10 Engineering and housekeeping commands received from AOSup

10.1 Engineering commands

This set of commands are used by the AOSup to request the TCS to execute specific actions. They are never used during an observation, but may be needed to perform engineering tasks, e.g.: for calibrations or maintenance operations.

Currently only commands to operate the hexapod are supported. They are listed in table 10.

10.2 Housekeeping commands

This set of commands are used for various housekeeping tasks such as modifying the logging level, forcing the logging of specific items into TCS logging system, enabling or disabling optional functions.

Housekeeping commands are listed in table 11.

[25] The AOSup is configured 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 waiting offload request except the most recent one, ensures that the TCS does not receive more offload commands than it can manage.

[26] Threshold and rescaling values are defined in a configuration file which is read at start time (see section 11).

Table 10: Hexapod commands issued by AOSup

Command	Description
HXPINIT	Initialize hexapod
HXPMOVETO	Move hexapod to absolute position
HXPMOVEBY	Move hexapod relative tom current position
HXPMOVSPH	Move hexapod on a sphere
HXPNEWREF	Set new reference point
HXPGETPOS	Get hexapod current position
HXPGETABS	Get hexapod absolute position
HXPOPENBRAKE	Open brake
HXPCLOSEBRAKE	Close brake
HXPISINITIALIZED	Test if hexapod has been initialized
HXPISMOVING	Test if hexapod is moving

Table 11: 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

10.2.1 LogItem

Requests to log some piece of information into the TCS Event System.

AOSup will has it's own logging system to be used for troubleshooting and engineering tasks. A selected subset of the log data can be stored in the TCS log system by means of `LogItem` commands[27].

10.2.2 LogOn/LogOff

Activate/deactivate the mirroring of AOS generated log messages to the `MsgD`.

After activation all the log messages[28] generated by the AOS other than being logged via the standard TCS logging mechanism, is also echoed to the `MsgD` in order to be merged with AOSup specific logs. This will help in the maintenance of the system by allowing to easily correlate the relevant log messages.

10.2.3 DbgLevel

Activate/deactivate the AOS log messages. This command may be used ad debugging tool to increase/decrease the verbosity of messages generated by AOS. Under normal conditions debug logging should be off (i.e.: `DbgLevel=0`).

[27] This feature is currently not used.
[28] Except for messages explicitly logged to TCS Syslog by the `LogItem` command.

Note that if a LogOn command has also been sent, all logs will be mirrored in the `MsgD` log file.

11 Configuration items

Table 12: AOS Configuration Parameters

Item	Type	Description
AOSLMsgDIp AOSRMsgDIp	string	Message daemon IP number. The IP address of the workstation running `MsgD` in the numerical dotted form.
AOSLtv_file AOSRtv_file	string	Basename of temporary file for the TV snapshot image. The procedure will create and update files named `<name>.dat` and `<name>.tmp`
AOSLfl_file AOSRfl_file	string	Basename of temporary file for the specification of the list of "flat" shapes available[29].
AOSLdbgLevel AOSRdbgLevel	int	AOS debug level. For debugging purposes AOS is provided of an internal log mechanism writing onto syslog. In normal use a debug level of 0 must be used; values 1 and 2 will provide increasingly verbose log.
AOSLsimulation AOSRsimulation	bool	Enable simulation mode when true. Simulation mode is used to perform a limited number of tests on AOS without the need to activate the AOSup.
AOSLlogdir AOSRlogdir	bool	AOS own log directory (currently unused)
AOSLtelEnable AOSRtelEnable	bool	Enable telemetry collection

A list of AOS related parameters[30] is shown in table 12. The naming follows a simple convention: they are all prefixed by either `AOSL` or `AOSR`, respectively, for left and right side.

AOS also uses data stored in two parameter files (for each side). They are stored in the AOS specific configuration directory[31] and contain data (thresholds, and conversion coefficients) related to the "Offload Modes" mechanism (see section 9) and for the `SetZernikes` command (see section 15.14).

Examples of the content of the mentioned files can be found in tables 13 and 14.

12 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.

[30] File: `.../aos/etc/aos.conf`
[31] Directory: `.../aos/configuration/AOS/`.

Table 13: `DXOffloadModes.dat` and `SXOffloadModes.dat`

```
# Gains and thresholds for Zernike values sent for Offlaod Modes commands
#
# 1. Thresholds.
# If no value exceeds the threshold, no correction is applied. If some do
# exceed the threshold, the correction is applied. Thresholds are applied
# before multiplying by the gains. As a consequence, thresholds values
# are in AO Supervisor units (meters).
#
#  z1    z2    z3    z4   z5   z6   z7   z8   z9  z10  z11  z12  z13  z14  z15  z16  z17  z18  z19  z20  z21  z22
  1.0  1.e-7 1.e-7 1.e-7 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1 5e-1

#
# 2. Gains.
# Zernike components are managed differently. Z1 is ignored,  Z2,Z3 must
# result in tip/tilt values in arcsec. Z4 must result in the focus value, in
# mm. All other values must result in standard zernike values in nanometers.
#
#  z1     z2       z3       z4     z5    z6   z7   z8   z9   z10  z11  z12  z13  z14  z15  z16  z17  z18  z19  z20  z21  z22
  0.0     -0.3e6   0.3e6    2e4    0     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

Table 14: `DXZernikes.dat` and `SXZernikes.dat`

```
# Gains for Zernike values sent by SetZernikes command
#
#  z1    z2    z3    z4   z5    z6    z7    z8    z9    z10   z11   z12   z13   z14   z15   z16   z17   z18   z19   z20   z21   z22
  0.0   0.0   0.0   0.0  1e-9  1e-9 -1e-9 -1e-9 -1e-9 -1e-9  1e-9  1e-9  1e-9  1e-9  1e-9  1e-9  1e-9  1e-9  1e-9  1e-9  1e-9  1e-9
```

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 AOSup.

12.1 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.

At the top of the panel you may find three items:

- **Snapshot button**: can be pressed to obtain a full dump of all AOS related `DD` variables onto the event logging system. It has been experimentally added to help in troubleshooting: when an unexpected behaviour or status is noticed by the TO, he/she should press the snapshot button so that current variable values are recorded for off-line analysis.
- **Command button**: enebles the Command Panel described in section 12.2.
- **Status info**: shows the status of the communication path with the AO Supervisor.

Below we may identify four main sections:

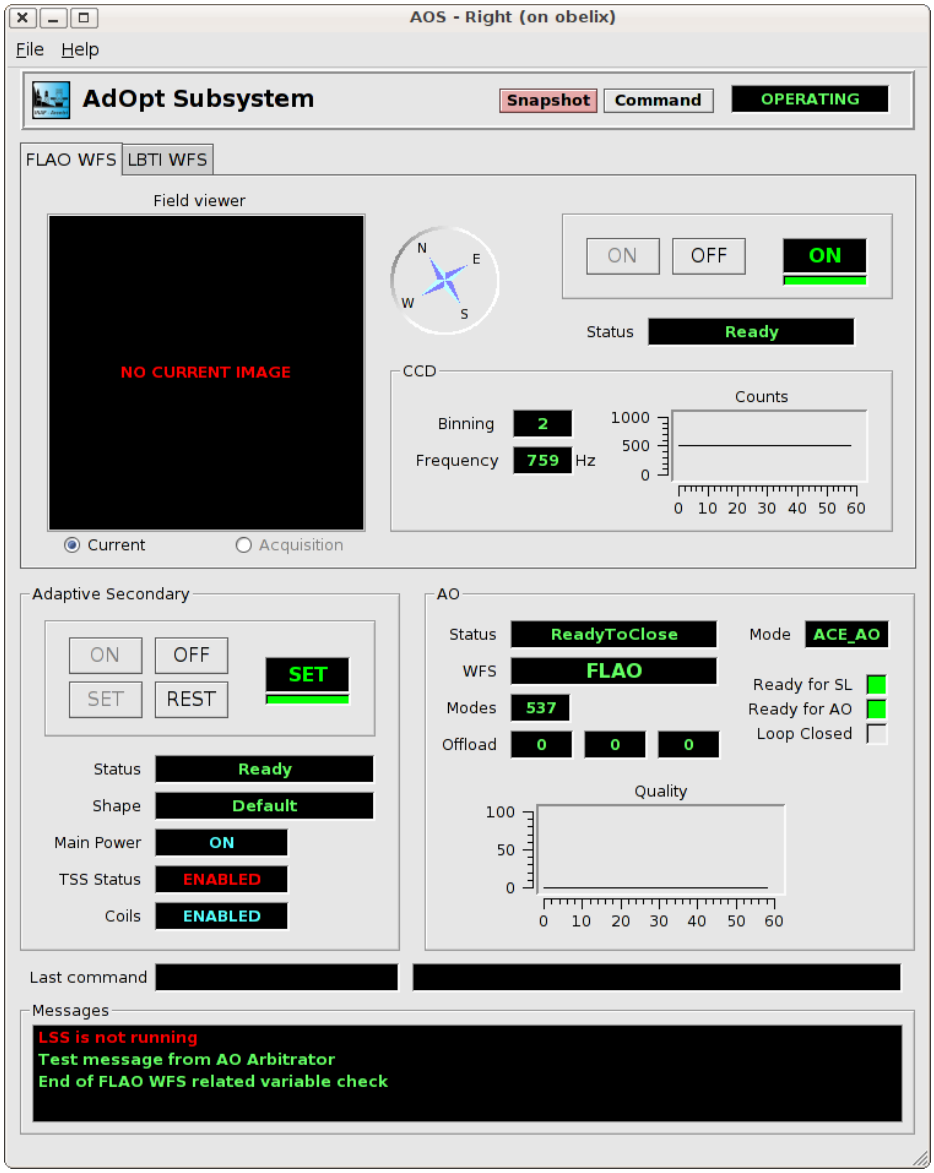

Figure 3: AOS GUI information panel

1. **Wavefront sensor section**, displaying data related to the Wavefront sensor. It is provided with two panels, one for the FLAO WFS and the otehr for the LBTI WFS. Each panel includes a snapshot of the technical viewer[32], 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

[32]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.

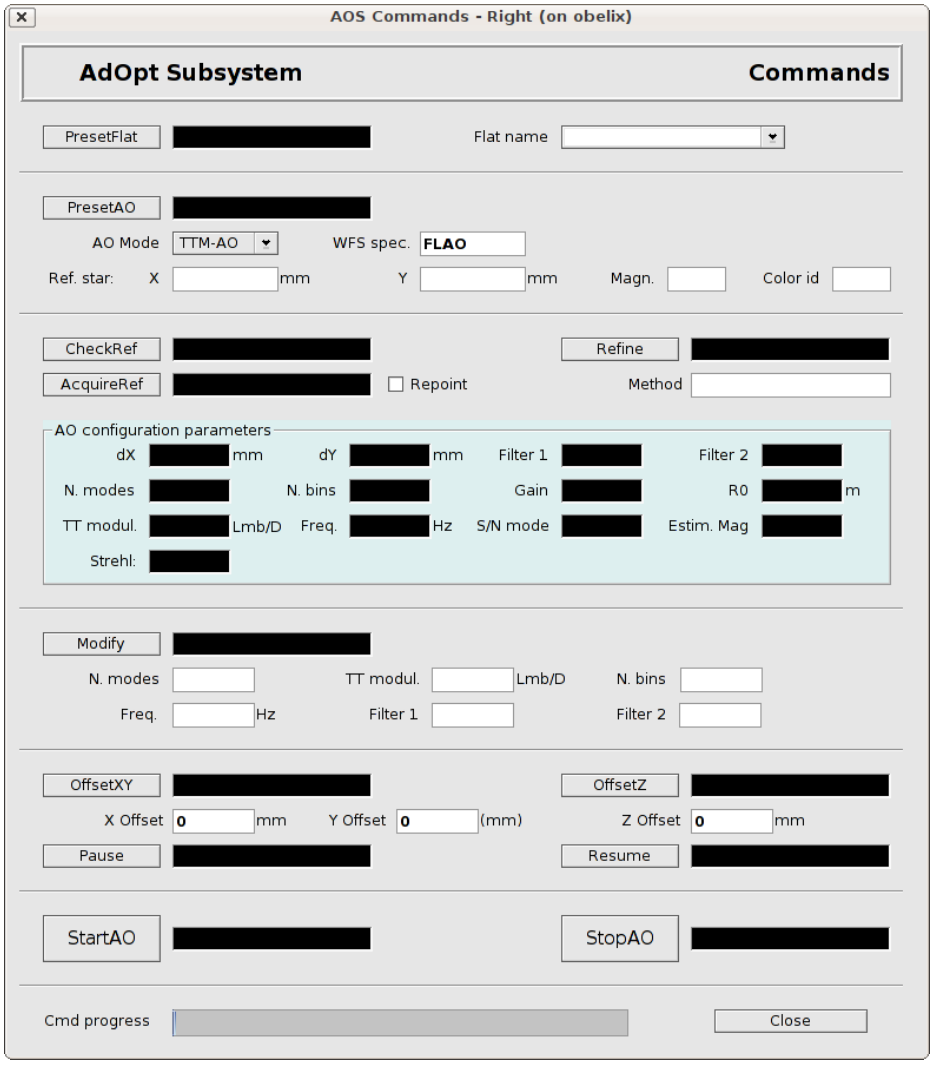

Figure 4: AOS GUI command panel

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[33]. It also include space for a temporal graph of some image quality indicator[34]. Three more on/off indicators provide some global status information.

4. **Log & Ctrl section**, containing items related to the management of the AOS subsystem. It will display status information, such as the current command in execution, messages and errors.

12.2 Command panel

Figure 4 shows a snapshot of the command panel available in the AOS GUI.

[33]The two tip-tilt components and the coma.
[34]Currently not implemented.

The purpose of the command panel is twofold.

It provides the capability to test the AOS and the interactions with the AOSup 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 13 through 15.

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 AOSup and any illegal or potentially dangerous command is simply rejected with an error message.

Part II

AOS Operational Commands

13 Introduction

In the following sections we will describe in full details the set of commands defined by AOS to allow the TCS to operate the AO System[35].

The main task of AOS is to translate each command received from TCS subsystems into a proper command (or sequence of commands) to be sent to the AO System for execution. For doing that AOS will obviously use the set of commands defined by the AO Arbitrator and already briefly described in section 1.2.

You may note that in most cases the translation between AOS commands and the corresponding AOArb commands is essentially a 1-to-1 mapping: the AOS command is directly implemented as a call to the corresponding AO Arbitrator command, sometimes after some adjustment of the arguments.

In a few cases the implementation of the AOS command is slightly more complex and may require intermediate operations before the AO Supervisor command is actually called.

Anyway, most AOS command have a corresponding AO Arbitrator command with the same name. In the following pages whenever a command name may be ambiguous, we use the prefix *AOArb:* to clearly identify commands to be sent to the AO Arbitrator.

All AO Arbitrator commands are executed in a synchronous way: after the call they will request the AO Supervisor to execute the related operations and will return after completion with either a success or failure status. Some command may also provide additional return values.

AOS Commands are defined as standard TCS commands by a specific interface definition[36].

The set of commands can be divided in two groups:

1. **Housekeeping commands**: which are provided to allow the TO to operate the AO System devices (switch on, switch off, etc.). These commands are listed in table 15.
2. **Observation commands**: which are provided to support the AO operation during an observation. These commands are listed in table 16.

AOS commands are described with many details in the following sections.

[35]The definition of the set of commands is the result of discussions with various people both from Arcetri and from Tucson: C. Biddick, N. Cushing, M. De La Pena, S. Esposito, R. Green, J. Kraus, D. Miller, A. Puglisi, A. Riccardi, P. Salinari, M. Wagner, with a special mention to J. Hill.

[36]See code in source file `.../aos/commands/AOSCommands.cpp`.

Table 15: AOS Housekeeping Commands

TCS comm.	AOArb comm.	Description
AdsecOn	AdsecOn	Set AdSec on
AdsecOff	AdsecOff	Set AdSec off
AdsecSet	AdsecSet	Set AdSec to status "set" (operating and with default shape)
AdsecRest	AdsecRest	Set AdSec to status "rest" (operating in safe position)
WfsOn	WfsOn	Set WFS on
WfsOff	WfsOff	Set WFS off
Snapshot		Dump values of DD relevant variables as events

Table 16: AOS Observation Commands

TCS comm.	AOArb comm.	Description
PresetFlat	PresetFlat	Preset AO System for seeing limited operation
PresetAO PresetAOg	PresetAO	Preset AO System for adaptive operation
AcquireRefAO	AcquireRefAO CheckRefAO	Acquire the reference star and become ready for closing the AO loop
CheckRefAO	CheckRefAO	Returns offset values between reference star actual and nominal position
RefineAO	RefineAO	Perform optimization of AO loop parameters
ModifyAO	ModifyAO	Modify some AO loop parameter
StartAO	StartAO	Start the AO mode (i.e.: close the AO loop)
OffsetXY OffsetXYg	OffsetXY	Offset AO pointing
OffsetZ	OffsetZ	Offset AO focus
CorrectModes	CorrectModes	Apply mirror shape correction
SetZernikes	SetZernikes	Apply mirror shape correction
Stop	Stop	Stop current operation
Pause	Pause	Temporarily suspend current operation
Resume	Resume	Resume suspended operation
setNewInstrument		Preset parameters related to an instrument

14 Housekeeping Commands

Housekeeping commands are intended to be used by the Telescope Operator to perform general operations on the AO System such as switching on and off the AO subsystems, and the like.

They are generally made available to the TO as "buttons" on the AOS GUI. Although it is possible, there is no reason for the IIF to provide access to these commands directly.

14.1 AdSecOn

The AdSecOn command directly invokes the *AOArb:*`AdSecOn` command to switch on the AdSec Subsystem. After receiving this command the AO System will switch on and initialize the Adaptive Secondary and put it in "safe" status; i.e.: ready for subsequent operations but with the thin shell in "safe" position[37].

The command has neither input arguments nor return values.

14.2 AdSecOff

The AdSecOff command directly invokes the *AOArb:*`AdSecOff` command to switch off the AdSec Subsystem. After receiving this command the AO System will put the thin shell in "safe" position and then switch off the hardware devices.

The command has neither input arguments nor return values.

14.3 AdSecSet

The AdSecSet command directly invokes the *AOArb:*`AdSecSet` command to put the thin shell in ready position[38]. After receiving this command the AO System will apply the default seeing limited shape to the shell.

The command has neither input arguments nor return values.

14.4 AdSecRest

The AdSecRest command directly invokes the *AOArb:*`AdSecRest` command to put the thin shell in "safe" position.

The command has neither input arguments nor return values.

14.5 WfsOn

The WfsOn command directly invokes the *AOArb:*`WfsOn` command switch on the WFS hardware.

The command has neither input arguments nor return values.

14.6 WfsOff

The WfsOff command directly invokes the *AOArb:*`WfsOff` command switch off the WFS hardware.

The command has neither input arguments nor return values.

[37]Safe position for the thin shell of the Adaptive Secondary means that it is "sucked" against the permanent magnets. In this state the mirror shape is not ready for seeing limited operations.

[38]This means that the Adaptive Mirror is ready for seeing limited operations. A precalibrated "flat" shape is actively maintained.

14.7 Snapshot

This command has no AoArb correspondance. When issued a selection of relevant variables are sent as TCS event messages to be logged. A corresponding button is provided for the operator. The idea is that the button is clicked by the TO when he/she notices some unexpected event, so that useful information can be later found int the log files to help understand what happened.

15 Observation Commands

15.1 PresetFlat

This command is issued to request the AO System to prepare itself for the FIX-AO[39] mode of observation. This mode doesn't require a reference star, but it is possible to select among several different precalibrated "flat" shapes depending on the instrument in use.

Note also that the AO system will always startup automatically and preset itself with a default flat shape even before it is connected to the AOS. So the PresetFlat command is only useful if a specific shape different from the default one has to be set.

Table 17: *AOArb:*`PresetFlat` command input arguments

Name	Type	Units	Comment
`FlatSpec`	string		Specification of the desired "flat"

Table 18: *AOArb:*`PresetFlat` command return values

Name	Type	Units	Comment
`FlatSpec`	string		Specification of the selected "flat"

PresetFlat command input arguments are specified in table 17.

The AOS will directly issue an *AOArb:*`PresetFlat` to the AOArb, then will wait for completion.

The `PresetFlat` command, when successful, returns a single value: the name of the selected flat shape (see table 18).

15.2 PresetAO / PresetAOg

This command comes in two flavors. The `PresetAO` version is intended to be used by instrument software (through the IIF), the `PresetAOg` version is suited to be called by the AOS GUI. In both cases the command is issued to prepare the AO System for an observation in adaptive mode, i.e.: one of TTM-AO, ACE-AO, ICE-AO[39].

The main difference between the two flavors is the way to specify the reference star position:

[39] See sect 1.2.2.

- **PresetAO**: requires the specification of the reference star by means of a `Position` object which contains all the details required to define the star (coordinates, magnitude, etc.).
- **PresetAOg**: requires the specification of reference star details by means of individual arguments separately (which are usually entered in corresponding field of the command panel of AOS GUI[40]).

From the implementation point of view when `PresetAO` is called the required values are extracted, or computed, from the `Position` object and then `PresetAOg` is called.

In `PresetAOg` the remaining arguments are added and then the function *`AOArb:`*`PresetAO` is called. Table 19 shows the input arguments of the function. Note that a number of input arguments are not mandatory and can have "null" values[41].

Table 19: *`AOArb:`*`PresetAO` command input arguments

Name	Type	Units		Comment
`AOMode`	string		Req.	Either "TTM-AO" or "ACE-AO" or "ICE-AO"
`Instr`	string		Req.	Specifies the instrument currently authorized (e.g.: `IRTC`)
`focStation`	string		Req.	Specifies the focal station currently authorized (e.g.: `bentGregorianFront`)
`WFS`	string		Req.	Specifies the source of WFS data(e.g.: FLAO)
`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[42]
`RotAngle`	real	radians	Req.	Angular position of rotator[a]
`GravAngle`	real	radians	Req.	Angular position of rotator with respect to gravity[a]
`Mag`	real	TBD	Opt.	Magnitude of reference star
`Color`	real	TBD	Opt.	Color Index of reference star
`R0`	real	TBD	Opt.	Estimated value of R0
`SkyBrghtn`	real	TBD	Opt.	Sky brightness
`WindSp`	real	TBD	Opt.	Wind speed
`WindDir`	real	TBD	Opt.	Wind direction

Upon receiving the command the AO System will perform all set up operations needed to prepare for the acquisition of a reference star. These include all the operations which can be done without having the light of the reference star available. The remaining part of setup will be requested with the `AcquireRefAO` command[43].

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, a command `AcquireRefAO` or `CheckRefAO` may follow.

[40] See section 12.

[41] The empty string is used as "null" value for string arguments and *NaN* is used as "null" value for numerical arguments.

[43] The sequence of low level operations needed to setup the AO System has been split in two steps (typically: `PresetAO` followed by `AcquireRefAO`) in order to allow the AO System to perform potentially time consuming adjustments (such as moving the mechanical assets of the WFS) while the telescope is slewing to point the source.

15.3 AcquireRefAO

The `AcquireRefAO` command is usually issued after a `PresetAO` in order to request the AO System to proceed to reference object acquisition.

The acquisition of the reference star[44] can be performed in two ways: 1) by moving the WFS internal stages properly, or 2) by adjusting telescope pointing. The `AcquireRefAO` command is thus provided with an argument to select the desired acquisition method.

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

1. An *AOArb:*`CheckRefAO`[45] command is issued to evaluate the amount of XY offset needed.
2. A command to adjust pointing is issued to PCS.
3. An *AOArb:*`AcquireRefAO` command is issued to actually perform reference star acquisition.

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

If telescope repointing is not wanted only step 3 above is performed. In this case, the pointing errors are compensated in the AO System by moving the WFS internal stages.

In either case the AOSup internal procedure, after putting the reference star in the right spot, will compute parameters needed for optimization of the AO loop and set up all the needed optical devices.

After successful completion, the *AOArb:*`AcquireRefAO` function will send back the computed AO loop parameters as detailed in table 20.

Table 20: *AOArb:*`AcquireRefAO` command return values

Name	Type	Units	Comment
ΔX	real	mm	Pyramid X displacement with respect to nominal position
ΔY	real	mm	Pyramid Y displacement with respect to nominal position
`slNull`	string		Selected slope null
`NModes`	int		Number of corrected modes
`Freq`	real	s	CCD Frequency
`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
`Strehl`	real		Measured Strehl ratio in J,H,K bands
`R0`	real	TBD	Measured R0
`MSNratio`	real[672]	TBD	Measured S/N per mode

[44] This operations essentially consists in putting the reference star light on top of the WFS pyramid.
[45] The `AOArb:CheckRefAO` command can be used also independently and is described in section 15.4.

After receiving the parameter block, the AOS will usually issue a `StartAO` command. In more complex cases the command might be followed also by a `RefineAO` or a `ModifyAO` command (see sections 15.5 and 15.6).

Note the ΔX and ΔY arguments. They return the amount of displacement of the WFS stages needed to put the source on top of the pyramid[46]: in this mode they can be used to evaluate the errors in the pointing model and, possibly, to correct it.

15.4 CheckRefAO

This command can be used to measure the offset between the reference star nominal and actual positions.

Table 21: *AOArb:*`CheckRefAO` command return values

Name	Type	Units	Comment
ΔX	real	mm	Pyramid X displacement with respect to nominal position
ΔY	real	mm	Pyramid Y displacement with respect to nominal position

It can be used to perform calibrations or to evaluate collimation parameters and it is not probably useful during usual observations, although it may be called as part of the reference star acquisition procedure described in section 15.3.

This command will simply issue an *AOArb:*`CheckRefAO` command to the AO Arbitrator. This will instruct the AOSup to take an image with the technical viewer, identify the reference star, which is supposed to be within the TV field, compute the ΔX and ΔY values and return them back.

After successful completion, the command returns the reference star position as detailed in table 21.

15.5 RefineAO

The `RefineAO` command[47] is used to request the AO system to perform better estimation of the AO loop parameters. It may be issued after the `AcquireRefAO` and will start the AOSup procedure[48] to optimize the selection of parameters.

Table 22: *AOArb:*`RefineAO` command input arguments

Name	Type	Units	Comment
`Method`	string		Optimization method

[46] Obviously in "repoint" mode the returned values should be zero.

[47] The `RefineAO` command has been defined fior future development, but the corresponding procedures have not yet been fully implemented in the AO Supervisor.

[48] The AO loop parameters optimization procedure evaluates the performances of the AO for a small interval of parameters values in order to determine the optimal set. It may require several minutes to complete, so it is offered as an optional function.

The command has a single argument which may be used to select a specific optimization method (see table 22).

This command can be issued only in reply to the successful completion of a previous *AOArb:*`AcquireRefAO` command. It is thus assumed that the reference object selected in the previous *AOArb:*`PresetAO` command is still correctly positioned.

Upon receiving the command AOSup will perform the optimization procedure and reply with the same parameter block described for the *AOArb:*`AcquireRefAO` command (see table 20).

15.6 ModifyAO

The `ModifyAO` command[49] has been defined to support the ICE-AO operating mode. It may be used to request the AO System 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 23).

Table 23: *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

This command can be issued only in reply to the successful completion of a previous AcquireRefAO and/or RefineAO command. It is thus assumed that the reference object selected in the previous `PresetAO` command is still correctly positioned.

AOSup will perform a check of the validity of parameters[50], recompute the optimization values, and will reply with the same parameter block described for the `AcquireRefAO` command (see sect. 15.3).

The calling procedure can in principle issue an arbitrary number of ModifyAO commands before requesting the closing of the AO loop with a StartAO command. The limitation being the capability of the telescope to track the reference star for the time needed for the operation.

15.7 StartAO

The `StartAO` command is issued by AOS to request the closing of the AO loop. It doesn't require any parameter in that it is only provided to synchronize the AO operation with the scientific instrument operation and when it is issued the AO System must be fully ready to close the AO loop.

AOS will simply call the corresponding *AOArb:*`StartAO` command and wait for a reply.

[49]The `ModifyAO` command has been defined but the corresponding procedures have not yet been fully implemented in the AO Supervisor.

[50]Parameter checking will be particularly strict for this command in order to ensure safe operation of the AO system.

15.8 OffsetXY / OffsetXYg

Also this command, as `PresetAO`, comes in two flavors. The `OffsetXY` version is intended to be used by instrument software (through the IIF), the `OffsetXYg` version is suited to be called by the AOS GUI.

The difference between the two flavors is in the way to specify the amount of offset:

- **OffsetXY**: requires the specification of the focal plane coordinates of the new pointing position.
- **OffsetXYg**: requires the specification of the offset value in mm in the focal plane reference.

When `OffsetXY` is used, the coordinates of the reference star got from the previous `PresetAO` command are subtracted from the coordinates provided as arguments and then `OffsetXYg` is called.

The AOS command `OffsetrXYg` in turn will issue an AOArb:`OffestXY` command to request the offsetting of the AO System (i.e.: move the WFS stages).

The AO arbitrator command requires two delta position values as detailed in table 24.

Table 24: *AOArb:*`OffsetXY` command input arguments

Name	Type	Units	Comment
`DeltaX`	real	mm	Requested X position offset
`DeltaY`	real	mm	Requested Y position offset

The command has different effects depending on the current mode. When the AO loop is closed any offset applied to the WFS stages will be corrected by the AO system and this will result in an offset of the field on the scientific camera[51]. When the AO loop is not closed (e.g.: after a `Pause` command) it will simply move the WFS stages.

15.9 OffsetZ

This command is issued in order to offset the focus position of the AO System (i.e.: by moving the WFS stages).

The AOS will directly call the corresponding *AOArb:*`OffsetZ` command.

If the command is issued when the AO loop is closed, it will be compensated by the AO system, thus it will result in a focus offset in the focal plane of the scientific camera; if the command is issued when the AO loop is open, it will simply move the Z stage of WFS.

The command requires one delta value as specified in table 25.

[51] Offsets in closed loop mode are compensated by AdSec with tip-tilt movements, which will need to be offloaded to telescope pointing correction by the mode offload mechanism (see section 9). Due to round-trip delays this means that only offsets of the order of about one half arcsec can be done in a single step. Bigger offsets must be either done in small steps with proper delay between them, or must be implemented by pausing the AO loop (see section 15.11) sending an OffsetXY and concurrently offsetting the telescope pointing, and then resuming the AO loop (see section 15.12).

Table 25: *AOArb:*`OffsetZ` command input arguments

Name	Type	Units	Comment
`DeltaZ`	real	mm	Requested focus offset

15.10 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 not available from the AOS GUI and can be issued only via the IIF.

A vector of Δ values must be specified (see table 26).

Table 26: *AOArb:*`CorrectModes` command input arguments

Name	Type	Units	Comment
`DeltaM`	real[672]	TBD	Modes correction vector

AOS will directly issue the corresponding AOArb command *AOArb:*`CorrectModes` and wait for completion.

15.11 Pause

This command temporarily suspend the AO loop; the AO System must remain ready to resume, i.e.: to close again the AO loop.

AOS will directly call the corresponding *AOArb:*`Pause` command and wait for completion.

AOSup will do the proper action (i.e.: open the AO loop) and then acknowledge the request. The command may be followed by either an `OffsetXY` or a `Resume` or a `Stop` command.

15.12 Resume

Resumes a suspended AO loop after a `Pause`. AOS will directly call the corresponding *AOArb:*`Resume` 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.

15.13 Stop

This command is issued to stop the current operation. After this command any setting defined by a previous `PresetAO` command will be canceled.

AOS will directly issue the corresponding *AOArb:*`Stop` command.

Table 27: *AOArb:*`Stop` command input arguments

Name	Type	Units	Comment
`Msg`	string		Reason for stopping

As shown in table 27, the command requires a string parameter containing a description of the reason for stopping (the argument is intended to provide the AO Supervisor with info related to the reason for stopping, e.g.: for use in the logging files).

15.14 SetZernikes

This command is used to apply a modal correction to mirror shape when the AO System is in Seeing Limited mode. The command is not available from the AOS GUI and can be issued only via the IIF.

AOS will directly call the corresponding *AOArb:*`SetZernikes` command. A vector of Δ values must be specified (see table 28).

Table 28: *AOArb:*`SetZernikes` command input arguments

Name	Type	Units	Comment
`DeltaM`	real[672]	TBD	Shape correction vector

This command is conceptually very similar to the `CorrectModes` command described in section 15.10, but is to be used when the AO System is operating in seeing limited mode to apply statical corrections to the mirror shape. It will actually result in very different operations at the AO System side[52].

15.15 setNewInstrument

This command has no AO Arbitrator counterpart. It is implemented because all TCS subsystem are required to be able to receive `setNewInstrument` commands.

Based on the specified instrument and focal station the AOS will preset internal variables which will be used to provide the required information when an *AOArb:*`PresetAO` command will be actually issued (see section 15.2).

The command will affect the following internal variables:

- `Instrument`: stores the authorized instrument identification string.
- `FocalStation`: stores the authorized focal station identification string.
- `WfsSpec`: based on specified instrument and focal station this variable is set to the correspondent WFS specification[53].

[52] In Seeing limited mode the corrections are applied to the mirror shape and involve only the AdSec. In Diffraction limited mode the correction are applied by modifying the "slope null" vector in the slope computer.

[53] This feature has been added to support different optical configurations, such as the one needed for LBTI operations.

A Source Files Identification

The main AOS related source files are listed and briefly described in table 29.

Table 29: AOS Source Files

`AOSAoApp.cpp` `AOSAoApp.hpp`	The core of the AOS. Together with code in `AOSHandlers` implements all the functionalities of AOS.
`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.
`AOSTelemetry.cpp` `AOSTelemetry.hpp`	Contains the code to manage the telemetry recording.
`AOS.cpp` `AOS.hpp`	Implementation of the AOS subsystem framework. It essentially provides for the management of the thread. The actual functionalities are implemented in `AOSAoApp`.
`AOSHandlers.cpp`	Contains the code for handling messages sent by the AOSup.
`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.

B Interface with AOSup

The code needed to interface AOS with the AOSup is contained in a directory subtree rooted in `../aos/aosupervisorlib`.

In other words the TCS source code tree contains all source files needed to properly compile and build the AOS executable.

Although modification of the interface is done with the greatest care, in order to avoid incompatibilities in the communication, it may be sometimes necessary to upgrade the interface code to align it with new features added to the AOSup code.

This is essentially a manual operation, because it might require changes in the organization of source files, anyway whenever modifications are limited to file content and no change in file organization has been made, a specific shell command (`update`) is included in the source code tree for updating the files. This anyway requires that the AOSup source tree is available and properly installed.

C Emulator and test procedures

In the AO Supervisor section of the source tree quoted in section B is contained a subdirectory (`../aos/aosupervisorlib/emulator`) which is not used for the building of AOS, but contains the source files needed to build two programs and some procedures which can be used for an offline test of the AOS. The purpose of the code contained there is to build an emulator and the required support process (`MsgD-RTDB`) which can take the part of the AO Supervisor to allow a preliminary test of the AOS when the actual AO Supervisor system is not available.

The building and usage of the AOSup emulator is described in a `README` file included in the set, which is copied here for your convenience:

```
This directory includes files needed to build an emulator of the AO Supervisor
to be used in software tests of AOS.

The Emulator is made up of three components:

msgdrtdb: a standalone version of the Message Daemon

thrdtest: a test program used to emulate the Ao Arbitrator

aostest.p: a command procedure to be fed to thrdtest to
emulate Ao Arbitrator operations.


BUILD PROCEDURE

The command 'make' issued on this same directory compiles and builds the two
executables: msgdrtdb and thrdtest.


RUNNING ENVIRONMENT

In order to properly communicate, the AOS must know the IP number of the
computer running msgdrtdb. This is specified in file 'lbt.conf' as parameter:
AOSLMsgDIp (left) and AOSRMsgDIp (right). The correct value must be set before
starting AOS.


HOW TO RUN THE TEST

1. Start TCS as usual


2. Start AOS and AOSGUI. The connection status indicator (top right on the GUI)
   shows: NO CONNECTION


3. start MsgD with:    source start_msgd.sh
```

```
   The connection status indicator on AOSGUI now shows: NO ARBITRATOR

   If the indicator remains in DISCONNECTED status, there is no communication
   between AOS and MsgD. You can verify the IP number actually used by AOS
   from the 'Help->About' panel in AOSGUI. The AO Supervisor IP number must
   be the one of the workstation where MsgD is running (usually: 127.0.0.1).


4. Start the test program with: ./thrdtest -aos right

   Use right of left as required.

   If the program starts correctly you should see the following:

AOARB.R: 6.6 [TH](Built: Oct 20 2010 12:13:51) - Dbg lev.:0, quiet, Line edit & history:No

Trying to connect to MsgD @ 127.0.0.1:9752

AOARB.R cmd:

   If you get a communication error, then MsgD is not running on this
   workstation.

   The prompt will show either AOARB.R or AOARB.L for right and left.


5. At the AOARB.x prompt start the command procedure with: exec aostest R

   Use "exec aostest L" for left side. The case is meaningful.

   You can now follow instructions and prompts from the procedure.

   NOTE: if you do not see the expected results during the execution of the test
   procedure, first verify you have specified the same side on all relevant
   commands.


NOTE: if the procedure is not executed to the end (e.g.: it has been interrupted
with CTRL-C) you must manually kill the message daemons. You can use the kill
command on the two processes named "msgdrtdb".
```

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: ?? ??? 201?
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