Revista Mexicana de Astronomía y Astrofísica Serie de Conferencias (RMxAC), **59**, 71–76 (2025)
© 2025: Instituto de Astronomía, Universidad Nacional Autónoma de México
https://doi.org/10.22201/ia.14052059p.2025.59.14

OBSERVATORY CONTROL SYSTEM AT SOUTH AFRICAN ASTRONOMICAL OBSERVATORY

S. Chandra[1,2,3], S. B. Potter[1], C. G. Gend[1], N. Erasmus[1], and R. Julie[1,4]

RESUMEN

El programa de Observatorio Inteligente (IO) del Observatorio Astronómico de Sudáfrica (SAAO) tiene como objetivo operar la mayoría de sus telescopios de forma robótica y desde un sistema de control centralizado equipado con tecnologías modernas para coordinar las observaciones de seguimiento de alertas proporcionadas por otros telescopios y observatorios tales como LSST, ROMAN, zTF, CTA, etc. Con los recientes desarrollos en el SAAO, el sistema de control del observatorio (OCS) ha demostrado ser un subsistema ya integrado de la compleja arquitectura del IO. El OCS, considerando sus distintos componentes: telescopios, instrumentos, observaciones como tal, registros, etc., facilitó a la arquitectura IO el poder integrar de manera única telescopios e instrumentos más antiguos, originalmente no diseñados para operaciones automatizadas. Al utilizar lenguajes populares como Python como componente principal, también resulta fácil cambiar según sean los requerimientos e igualmente el poder comunicarse mediante robots. También se ha reducido gran parte de la carga del equipo de gestión del observatorio al proporcionar una base de datos con un protocolo de comunicación fácil para la gestión y visualización de los datos.

ABSTRACT

The intelligent observatory (IO) programme at SAAO aims to operate most of its hosted telescopes robotically from a centralized control system equipped with modern technologies to coordinate the followup observations triggered by the most sophisticated global facilities like LSST, ROMAN, zTF, CTA etc. With the recent developments at SAAO, the observatory control system (OCS) has proven to be an integrated sub-component of the complex IO architecture. The OCS, because of a simplistic fragmentation in terms of the definitions of the various components: such as telescopes, instruments, observations, logging; helped the IO architecture uniquely to integrate very old telescope and instruments, originally not designed for the automated operations. Using popular languages like Python as a major component, it also become easy to change as per need and also to communicate using robots. It has also reduced a lot of burden of the observatory management team by providing a communicable database for managements and data visualization

Key Words: IO Architecture — Observatory: Observatory Control System

1. GENERAL

In the current era of the multi-messenger astronomy the coordinated multi-wavelength campaigns have become an effective tool for a wide range of astronomical research. Being either variable stars in our galaxy, outbursts in galactic compact objects, active galaxies, or distant gamma-ray bursts, etc., wherever variability is a crucial characteristic, the coordinated monitoring provides only ways to uncover a complete picture of the system and underlying physical processes. Depending on the astrophysical systems the campaigns may become very time critical and to involve real-time coordination between various international observing facilities from ground and the space. A well structured campaign with real-time feedback system has served backbone for reporting ice-breaking discoveries such as kilonovae as the optical counterparts to some gravitational wave events (Abbott et al. 2017), and the first electromagnetic (EM) counterparts of neutrino sources (IceCube Collaboration et al. 2018; Andreoni et al. 2024). The coordinated campaigns have historically helped many traditional research fields to uncover deep hidden details of the physical processes (Plucinsky et al. 2017; Yao et al. 2021; EHT MWL Science Working Group et al. 2021). The upcoming mega facility Vela Rubin Observatory or LSST (Ivezić et

[1]South African Astronomical Observatory (SAAO), 1 Observatory Road, Observatory, Cape Town, 7925, South Africa (sunil.chandra355@gmail.com).

[2]Centre for Space Research, North-West University, Potchefstroom Campus, 11 Hoffman St., Potchefstroom, 2520, North-West, South Africa (chandra@saao.ac.za).

[3]Physical Research Laboratory, Mt. Abu Observatory, Mount Abu, Rajasthan, 507501, India (schandra@prl.res.in).

[4]South African Radio Astronomy Observatory, 2 Fir Street, Cape Town, 7925, South Africa.

al. 2019) shall be streaming millions of triggers every night with many new exotic unknown breeds of astrophysics systems.

For example, characterizing and classifying optical transients or electromagnetic counterparts of neutrinos and gravitational waves, variable nature of broad-band emitters pose a huge challenge of covering entire accessible energy spectrum with best possible simultaneity. These coverage may need many repetitions in similar of different orders based on the requirements of the scientific objectives and telescope times. Such complex observing plans not only demand rapid actions but also a nearly real-time communications between different participating facilities or collaborations. Intelligent observatory (IO) programme[5](Potter 2021) is an effort to upgrade the existing observing facilities at SAAO in Sutherland to robotize and convert the South African plateau as a rapid followup machine in the LSST era. The IO's flexible architecture (van Gend et al. 2020) has provisions of switchable robotic and manual modes of telescopes operations. This also enables a centralized computer brain for communicating with the observer (or robots) and multiple observing facilities. The IO architecture can be split between three major components **1) The Local Control Unit (LCU):** set of modules to communicate with telescope control, instrument control and auxiliary services attached to the instrument (guider, focusser etc), **2) IO Observatory Control System (IO OCS or OCS):** modules to enable **Application Programming Interface (API)** driven communications between observer and the observatory, and **3) The IO Interfacing Layer (IOIL):** set of modules converting the instructions from the OCS to the frameworks of individual telescopes-instruments pairs. The LCUs are very different for the instrument and telescopes combinations. The IO team has developed layers (e.g., thrift or python wrapper) to communicate original control software (cameras, filter-wheels, telescope drives, dome drives etc) and avoid re-development from scratch, where-ever possible (Erasmus, N., et. al., 2024). The IOIL are mostly Python libraries developed in-house to establish a general format communication, preferably by converting the incoming instructions in the JSON format (e.g., `schedule_poller` at SAAO) to small scripts with settings forwarded to the LCUs. The OCS as per the design, is very important component of the IO architecture (refer to Figure 1) because this aims to provide easy, descriptive and robotic two-way communications, support multi-platform access

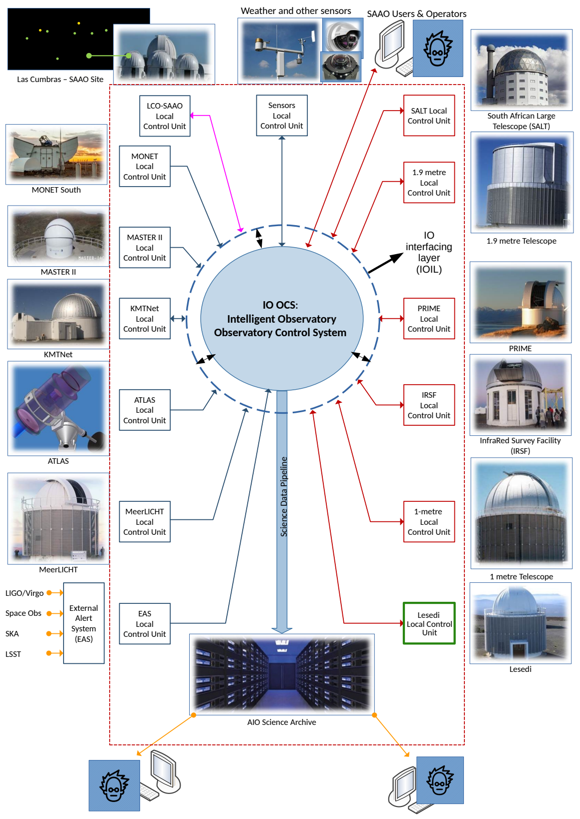

Las Cumbras – SAAO Site
Weather and other sensors
SAAO Users & Operators
South African Large Telescope (SALT)
1.9 metre Telescope
PRIME
InfraRed Survey Facility (IRSF)
1 metre Telescope
Lesedi
MONET South
MASTER II
KMTNet
ATLAS
MeerLICHT
LCO-SAAO Local Control Unit
Sensors Local Control Unit
SALT Local Control Unit
MONET Local Control Unit
1.9 metre Local Control Unit
MASTER II Local Control Unit
IO interfacing layer (IOIL)
KMTNet Local Control Unit
IO OCS: Intelligent Observatory Observatory Control System
PRIME Local Control Unit
ATLAS Local Control Unit
IRSF Local Control Unit
Science Data Pipeline
MeerLICHT Local Control Unit
1-metre Local Control Unit
LIGO/Virgo
Space Obs
SKA
LSST
External Alert System (EAS)
EAS Local Control Unit
Lesedi Local Control Unit
AIO Science Archive

Fig. 1. The high-level design of the Intelligent Observatory Architecture. The IO OCS or OCS serves as the central server which communicates the users/admin and local control units (LCUs). The communication with the LCUs are mediated through IO interfacing layer. The green box around Lesedi telescope indicates that it is already operated in a robotic mode.

(API controlled), and easy to be integrated with other systems and layers. This manages various databases and serves as the brain of the IO system at SAAO.

2. OBSERVATORY CONTROL SYSTEM (OCS) AT SAAO

The OCS at SAAO (henceforth referred as OCS, mentioned otherwise) primarily consists of the modules developed by the Las Cumbres Observatory (LCO), with the same name[6], enabling the networking of 25 telescopes of the LCO network in the different geographical locations. A subset of publicly available modules with a minimal optimization and added layering for the networking using Python, in close collaborations with the development team at

[5]https://io.saa.ac.za

[6]https://observatorycontrolsystem.github.io

VII Workshop on Robotic Autonomous Observatories (October 16-20, 2023)
Editors: Maria Gritsevich, Alberto J. Castro-Tirado, Petr Kubánek, Shashi B. Pandey, and David Hiriart - DOI: https://doi.org/10.22201/ia.14052059p.2025.59.14

LCO, serves as the brain of the OCS. Having a generalized structure, vast ranges of the basic functionalities, robotic communications for the admin and user related activities, network based communications for almost internal and external activities, the OCS is very easy to add more intelligent layers to the IO structure in-near future such as weather based decision making, analysis pipelines, and artificial intelligence (AI) driven characterization, cataloging and alerts etc. The adoption of the LCO's modules are integrated in such a way that the future upgrades of these can be easily used without any major change of other communicating layers. These justifies the use of the local production of the LCO's modules a best suited publicly available resource to serve the vision of IO at SAAO.

2.1. *Modules adopted from the LCO's OCS*

The recent versions of the following modules make the complete OCS being used for the IO:

- `configdb:`[7] This is Django, HTML and Python based module to create, maintain and communicate the database related to the site, telescope, instruments, detectors, and optical components. The module, because of defining the sub-components of the instruments at a very fundamental level of information (with additional controls in the form of validation schema), is capable of creating a complex and shared telescope-instrument combinations. The database used by this is freely available (`PostgreSql` and hence adds to its applicability for the general use. The `configdb` database (see Figure 2 for different sub-components and the flow-chart) stores information in a multi-layered JSON format which is easily callable for the API requests through GET instances. Figures 3 and 4 refer to the snapshots of various instances in the JSON format. The customization of the this database to apply the local requirements of the SAAO telescopes and instruments are done through using generic modes, optical path elements, and specific custom validation_schema. We also had defined several specific 'tags' while defining the 'instrument' and 'instrument type' to apply many complex settings and availing those to the users. These tags are interpreted by IIL in a comprehensive and machine readable instructions before forwarding it to the LCUs.

- `downtime:` A library to create, manage and communicate a database of downtime of the telescope and instruments. This database enable telescope operation team to keep records of the duration of unavailability of telescope-instrument pairs (due to maintenance, upgradation or technical glitches) and which is accessed by the operational modules in real-time manner through network. This module is used at SAAO in its original format without any modification. It is used to disable the telescopes from the robotic network if it is used for the manual operation, student training programs, and the maintenance.

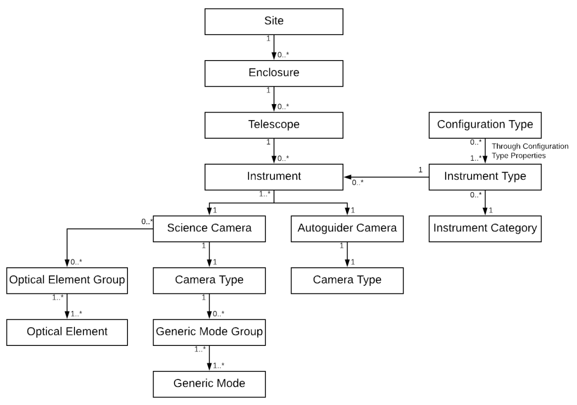


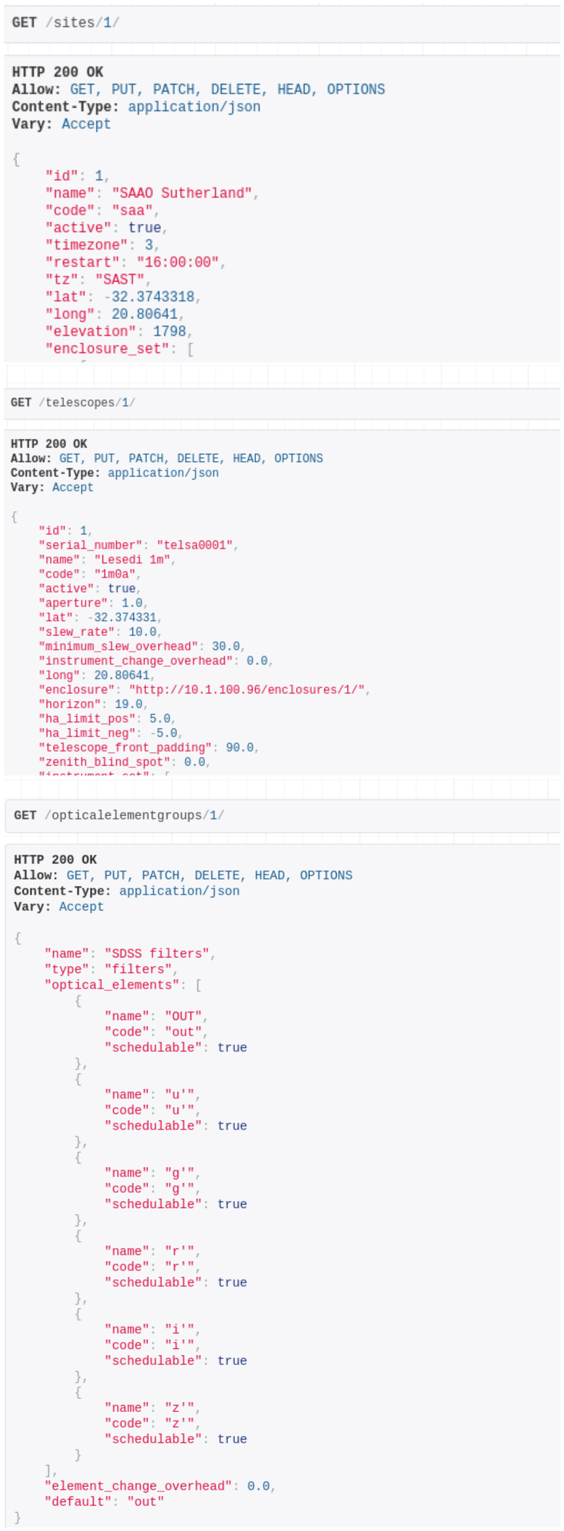
Fig. 2. The schematic drawing, borrowed from the LCO official page, showing different components used in the configdb database. It also represent a flow of defining the database. These basic components are used for the formation of the telescope-instrument pairs.

- `observation_portal:`[8] This library (Python and HTML) is the central package managing an API driven portal to create semester, collaboration, proposal, configurations, and many other useful tasks related to the operations for any observatory. It also has capabilities of sending out automated notifications for the new proposal calls, status of an active proposal, and communication between telescope control about the status of the data acquisition etc. It generates and maintain a comprehensive database which can be accessed for automated data archiving and visualization. In order to meet the compatibility of the SAAO's observing facilities, telescope operation guidelines, and user requirements, the first level of customization was made through applying 'validation_schema' and by defining specific 'configurations' and without making any actual changes to

[7]https://observatorycontrolsystem.github.io/components/configuration_database/

[8]https://observatorycontrolsystem.github.io/components/observation_portal/

VII Workshop on Robotic Autonomous Observatories (October 16-20, 2023)
Editors: Maria Gritsevich, Alberto J. Castro-Tirado, Petr Kubánek, Shashi B. Pandey, and David Hiriart - DOI: https://doi.org/10.22201/ia.14052059p.2025.59.14

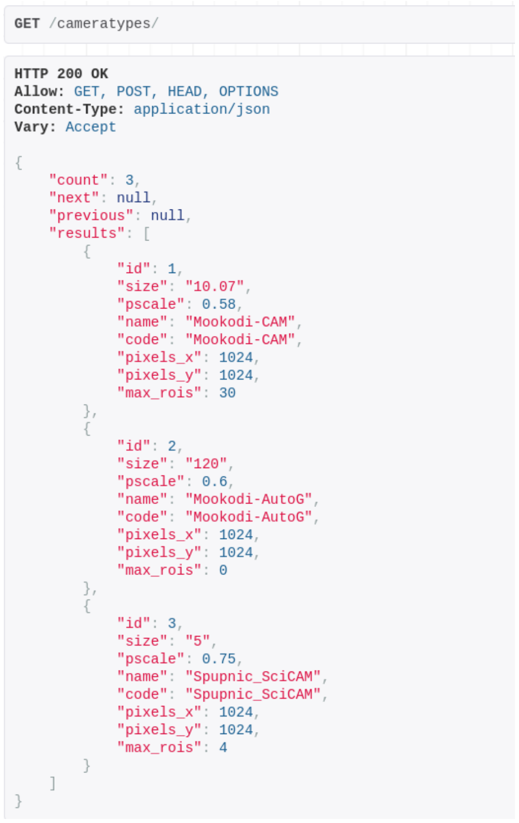


```
GET /sites/1/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "name": "SAAO Sutherland",
    "code": "saa",
    "active": true,
    "timezone": 3,
    "restart": "16:00:00",
    "tz": "SAST",
    "lat": -32.3743318,
    "long": 20.80641,
    "elevation": 1798,
    "enclosure_set": [
```

```
GET /telescopes/1/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "serial_number": "telsa0001",
    "name": "Lesedi 1m",
    "code": "1m0a",
    "active": true,
    "aperture": 1.0,
    "lat": -32.374331,
    "slew_rate": 10.0,
    "minimum_slew_overhead": 30.0,
    "instrument_change_overhead": 0.0,
    "long": 20.80641,
    "enclosure": "http://10.1.100.96/enclosures/1/",
    "horizon": 19.0,
    "ha_limit_pos": 5.0,
    "ha_limit_neg": -5.0,
    "telescope_front_padding": 90.0,
    "zenith_blind_spot": 0.0,
```

```
GET /opticalelementgroups/1/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "name": "SDSS filters",
    "type": "filters",
    "optical_elements": [
        {
            "name": "OUT",
            "code": "out",
            "schedulable": true
        },
        {
            "name": "u'",
            "code": "u'",
            "schedulable": true
        },
        {
            "name": "g'",
            "code": "g'",
            "schedulable": true
        },
        {
            "name": "r'",
            "code": "r'",
            "schedulable": true
        },
        {
            "name": "i'",
            "code": "i'",
            "schedulable": true
        },
        {
            "name": "z'",
            "code": "z'",
            "schedulable": true
        }
    ],
    "element_change_overhead": 0.0,
    "default": "out"
}
```

Fig. 3. The snapshots of the 'site' (top), 'telescope' (middle), and 'optical elements': filters, slits, grating, half-wave plates etc (bottom) stored in the database and accessible through the observation_portal API.



```
GET /cameratypes/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 3,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "size": "10.07",
            "pscale": 0.58,
            "name": "Mookodi-CAM",
            "code": "Mookodi-CAM",
            "pixels_x": 1024,
            "pixels_y": 1024,
            "max_rois": 30
        },
        {
            "id": 2,
            "size": "120",
            "pscale": 0.6,
            "name": "Mookodi-AutoG",
            "code": "Mookodi-AutoG",
            "pixels_x": 1024,
            "pixels_y": 1024,
            "max_rois": 0
        },
        {
            "id": 3,
            "size": "5",
            "pscale": 0.75,
            "name": "Spupnic_SciCAM",
            "code": "Spupnic_SciCAM",
            "pixels_x": 1024,
            "pixels_y": 1024,
            "max_rois": 4
        }
    ]
}
```

Fig. 4. The API instance of the list of science cameras loaded in the database at SAAO.

the codes. Another advanced level of customization is also possible via overriding any serializer in the project (observation portal project by LCO with forking options), or overriding the as_dict methods of models which are used for generating API responses. The detailed example and methods can be found on the official page of the LCO's project. We may need to adapt more specific customization using overrides for the integration of old telescopes to the IO network. Note that the telescopes we are integrating with the IO network, were not originally meant for the robotic modes of operations.

The SAAO's implementation of these packages are made using dockers, hosted in the virtual machines at different servers at SAAO headquarter in Cape Town, South Africa. The APIs in the dockers are accessed through their respective IPs and ports (for example http://10.1.100.96:7000 for the `configdb` service). The 'api' and 'admin' components are accessible by using these tags in the front of the IP:PORT combination. The Django web framework server based centralized authentication (`OAUTH` scheme hosted at the observation portal database) is

VII Workshop on Robotic Autonomous Observatories (October 16-20, 2023)
Editors: Maria Gritsevich, Alberto J. Castro-Tirado, Petr Kubánek, Shashi B. Pandey, and David Hiriart - DOI: https://doi.org/10.22201/ia.14052059p.2025.59.14

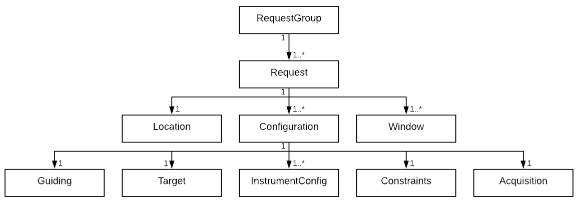
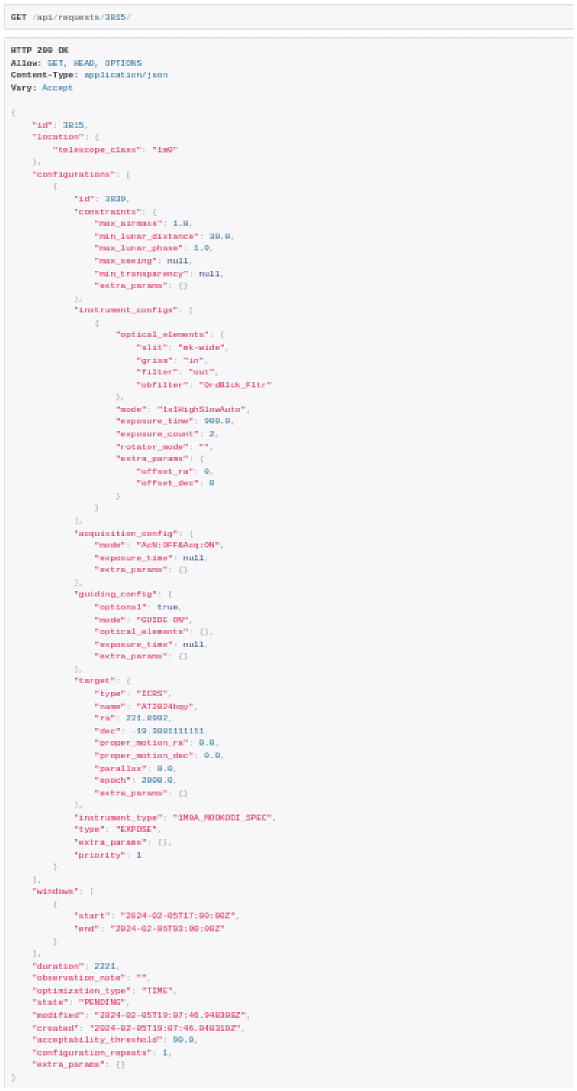


Fig. 5. The schematic borrowed from the official page by the LCO describing the hierarchy of properties of a RequestGroup. These requests can be made through the submission forms or programmatically using the API.



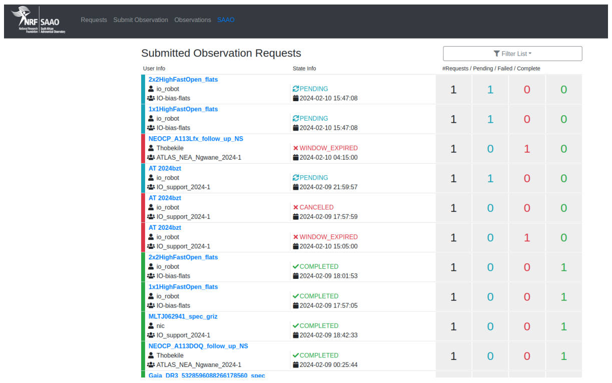
```
GET /api/requests/3815/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 3815,
    "location": {
        "telescope_class": "1m0"
    },
    "configurations": [
        {
            "id": 3839,
            "constraints": {
                "max_airmass": 1.8,
                "min_lunar_distance": 30.0,
                "max_lunar_phase": 1.0,
                "max_seeing": null,
                "min_transparency": null,
                "extra_params": {}
            },
            "instrument_configs": [
                {
                    "optical_elements": {
                        "slit": "mk-wide",
                        "grism": "in",
                        "filter": "out",
                        "obfilter": "OrdBlck_Fltr"
                    },
                    "mode": "1x1HighSlowAuto",
                    "exposure_time": 900.0,
                    "exposure_count": 2,
                    "rotator_mode": "",
                    "extra_params": {
                        "offset_ra": 0,
                        "offset_dec": 0
                    }
                }
            ],
            "acquisition_config": {
                "mode": "AcN:OFF&Acq:ON",
                "exposure_time": null,
                "extra_params": {}
            },
            "guiding_config": {
                "optional": true,
                "mode": "GUIDE ON",
                "optical_elements": {},
                "exposure_time": null,
                "extra_params": {}
            },
            "target": {
                "type": "ICRS",
                "name": "AT2024bgy",
                "ra": 221.8992,
                "dec": -19.3881111111,
                "proper_motion_ra": 0.0,
                "proper_motion_dec": 0.0,
                "parallax": 0.0,
                "epoch": 2000.0,
                "extra_params": {}
            },
            "instrument_type": "1M0A_MOOKODI_SPEC",
            "type": "EXPOSE",
            "extra_params": {},
            "priority": 1
        }
    ],
    "windows": [
        {
            "start": "2024-02-05T17:00:00Z",
            "end": "2024-02-06T02:00:00Z"
        }
    ],
    "duration": 2221,
    "observation_note": "",
    "optimization_type": "TIME",
    "state": "PENDING",
    "modified": "2024-02-05T19:07:46.948308Z",
    "created": "2024-02-05T19:07:46.948210Z",
    "acceptability_threshold": 90.0,
    "configuration_repeats": 1,
    "extra_params": {}
}
```

Fig. 6. The snapshot of a submitted request structured in JSON format.

used for managing the user accounts for all the modules. Various important tasks from different services such as, admin: management of users & proposals, adding new telescopes, instruments, changing status of existing pairs, etc; and users: registration, submitting new observing requests, checking the status of submitted requests, getting proposal metric etc; can easily be performed by sending requests to corresponding APIs either in batch mode or pragmatically as part of alert management systems. All these modules are equipped with interactive python shells preloaded with a list of useful libraries for debugging, updates etc. Within the OCS structure: the `configdb` and `downtime` runs the databases of the instruments and their availability and they are communicated in realtime using internal network communications. The `observation_portal` runs all the API back-ends related to the members, observation requests, proposals, logs, schedule information, status record of submitted requests etc. For any communication to `observation_portal`, it sends queries to `configdb` and `downtime` to verify and framing any observing request. Any submitted request, for example the one shown in Figure6, can be broken into different parts as shown in Figure 5. The LCUs, hosted at the computers controlling the different telescopes and instruments, are communicated through two way API requests by the OCS as soon the schedule is uploaded on the OCS server. In order to schedule the submitted requests we are using a python module '`adaptive_scheduler`[9] developed by the LCO team. This scheduler uses Google's OR-Tools[10] as kernel and avails configurations to support the SCIP, CBC and GLPK free algorithms. It also supports the optimization using premium services like GUROBI[11], but based on our local traffic of the requests, the free algorithms serves the purpose.

The robotic communications sent form the `io_robot`, either as direct submissions (calibrations: bias, flats & arcs) or regular submission (transient monitoring from the Transient Name Server (TNS) triggers after quality filtering) are already tested and being used since last two trimesters. The `io_robot`, which is an auxiliary service replacing human interventions at the top level with pre-defined setup, currently filters triggers from both the TNS and GCN and submits requests to the IO, as applicable. We are testing other trigger resources like BAWG[12] and FINK[13] to enable full compliance for following up observations of the triggers from the LSST. A more technical paper on IO OCS is in preparation

[9]`https://github.com/observatorycontrolsystem/adaptive_scheduler`
[10]`https://developers.google.com/optimization`
[11]`https://www.gurobi.com/`
[12]`https://t.me/+lfchnd4klOdmYTYy`
[13]`https://fink-broker.org/`



Submitted Observation Requests

User Info	State Info	#Requests / Pending / Failed / Complete			
2x2HighFastOpen_flats io_robot IO-bias-flats	PENDING 2024-02-10 15:47:08	1	1	0	0
1x1HighFastOpen_flats io_robot IO-bias-flats	PENDING 2024-02-10 15:47:08	1	1	0	0
NEOCP_A113Lfx_follow_up_NS Thobekile ATLAS_NEA_Ngwane_2024-1	WINDOW_EXPIRED 2024-02-10 04:15:00	1	0	1	0
AT 2024bzt io_robot IO_support_2024-1	PENDING 2024-02-09 21:59:57	1	1	0	0
AT 2024bzt io_robot IO_support_2024-1	CANCELED 2024-02-09 17:57:59	1	0	0	0
AT 2024bzt io_robot IO_support_2024-1	WINDOW_EXPIRED 2024-02-10 15:05:00	1	0	1	0
2x2HighFastOpen_flats io_robot IO-bias-flats	COMPLETED 2024-02-09 18:01:53	1	0	0	1
1x1HighFastOpen_flats io_robot IO-bias-flats	COMPLETED 2024-02-09 17:57:05	1	0	0	1
MLTJ062941_spec_griz nic IO_support_2024-1	COMPLETED 2024-02-09 18:42:33	1	0	0	1
NEOCP_A113DOQ_follow_up_NS Thobekile ATLAS_NEA_Ngwane_2024-1	COMPLETED 2024-02-09 00:25:44	1	0	0	1
Gaia_DR3_5328596088266178560_spec					

Fig. 7. The snapshot of the IO front-end (`https://ocsio.saao.ac.za`) showing a list of submitted requests. A mojor part of the design of this front-end is taken from the OCS-EXAMPLE[15] project by the LCO.

for the Journal of Astronomical Telescopes, Instruments, and Systems (JATIS).

3. WORK-FLOW AND SUMMARY

The demands of the IO architecture to serve a centralized brain for communicating the users and LCUs are best met by the current verison of the OCS at SAAO. The local production of the LCO's modules, because of its flexibility, have been a vital components of the OCS. The telescope operations (TOPS) and telescope time allocation committee (TAC) at SAAO are responsible for managing the proposal submission and time allocation for any trimester. The respective proposer(s) need to register at IO frontend[16]. Any user with already approved instrument time, submits observation requests to their assigned proposals. The submission can be made through the said portal or programmatically. The requests framed in formats supported by the pre-defined validation schema are validated in realtime and only the successful ones are submitted. The submitted requests are stored in the `observation_portal` database. The `adaptive_scheduler` runs every 2 minutes to generate an optimized schedule for all active instruments and update it on the OCS API. The IOIL analyses the schedules on a regular interval to check for any urgent time-critical or rapid response submissions. It can cancel the ongoing monitoring, if urgent action is needed otherwise before every new telescope pointing the most updated schedule is used. For every pointing the IOIL communicates with the LCUs for observing instructions and also to get updates about the data acquisition. It is also responsible for updating the status tags at the OCS server such as 'PENDING', 'ATTEMPTED' and 'COMPLETED' with a short summary. Once the observation is tagged as 'COMPLETED', a trigger is generated for the real-time analysis procedures (pipelines) and the SAAO Science Archive to ingest the data. The pipelines and archiving system are under commissioning phase.

Figure 7 shows a glimpse of the IO front-end showcasing the active IO system, listing various requests from manual and robotic submissions.

Acknowledgements: The IO programme is supported by the National Research Foundation, South Africa. SC acknowledges the discussions with Mr. Jon Nation from the LCO which were very helpful in adapting the OCS into IO architecture. SC also acknowledges the discussions with Dr. Christain Hettlage and Mr. Iriwaan Simon related to debugging of networking the different dockerized services. The authors also acknowledge entire efforts of the LCO team for this Open Source Observatory Control System Project, which is supported with the generous financial support from the Heising-Simons Foundation.

REFERENCES

Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2017, ApJ, 848, L12, doi:10.3847/2041-8213/aa91c9
Andreoni, I., Coughlin, M. W., Criswell, A. W., et al. 2024, APh, 155, 102904, doi:10.1016/j.astropartphys.2023.102904
EHT MWL Science Working Group, Algaba, J. C., Anczarski, J., et al. 2021, ApJ, 911, L11, doi:10.3847/2041-8213/abef71
Erasmus, N., Potter, S. B., van Gend, C. H. D., et al. 2025, RMxAC, 59,
van Gend, C. H. D. R., Sickafoose, A. A., Potter, S. B., et al. 2020, Proc. SPIE, 11452, 1145206, doi:10.1117/12.2562161
IceCube Collaboration, Aartsen, M. G., Ackermann, M., et al. 2018, Sci, 361, 147, doi:10.1126/science.aat2890
Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, ApJ, 873, 111, doi:10.3847/1538-4357/ab042c
Potter, S. 2021, AFAS2 2021: AfAS, 34
Plucinsky, P. P., Beardmore, A. P., Foster, A., et al. 2017, A&A, 597, A35, doi:10.1051/0004-6361/201628824
Yao, Y., Kulkarni, S. R., Burdge, K. B., et al. 2021, ApJ, 920, 120, doi:10.3847/1538-4357/ac15f9

[16]`https://ocsio.saao.ac.za`