The Condor Array Telescope Data API

Document 0004-A

June 2021

Contents

1 Introduction **4**

2 Overview **4**

3 Obtaining an Authorization Token **4**

4 The /api/exposures/ Endpoint **5**

4.1 General Usage 5

4.2 Supplying Additional Parameters to the Endpoint 6

4.3 Endpoint Parameters 6

4.4 Raw Database Field Parameters 7

4.5 URL Encoding 7

5 The /api/exposures/<filename> Endpoint **8**

1 Introduction

This document describes the Condor Array Telescope data API. The data API provides an interface to the Condor Array Telescope database and data storage, allowing records to be created, viewed, and updated. This document specifically focuses on using the API to view records, from the perspective of a user seeking to process and analyze observations.

2 Overview

Accessing the Condor Array Telescope data API involves several steps:

1. If you do not already have a user account on the Condor Array Telescope web site, then sign up for one by visiting
   - https://condorarraytelescope.org/account/signup/.
2. Request that your user account be added to the Condor group. (Currently, the only way to do this is by emailing kenneth.lanzetta@stonybrook.edu, providing your user account user name in the email.)
3. Use the API to obtain an access token following instructions described in § 3 below.
4. Use the API to view records following instructions described in §§ 4 through 6 below.

3 Obtaining an Authorization Token

Access to the Condor Array Telescope data API is restricted using token authentication. Hence in order to use the API, you will first need to obtain an access token by making an HTTP POST request containing your user name and password to

- https://condorarraytelescope.org/api/token-auth/.

This is easily accomplished in python using the "requests" package via

```
import requests

username = 'myusername'
password = 'mypassword'

data = {'username': username, 'password': password}

result = requests.post(
  'https://condorarraytelescope.org/api/token-auth/',
  json=data
)
print(result.text)
```

There are two things to note here: (1) You must, of course, substitute your user name for "myusername" and your password for "mypassword." And (2) you must, of course, have the python "requests" package installed on your computer for this to work. This will print a result similar to

```
{"token":"7304045ab7882812aa50c403a17f36c9cc3d1c8f"},
```

which contains your 40-digit access token. Your access token provides access to the API, and you should treat it like you would any other credential, keeping it secret and safe.

What is the point of using token authentication, if ultimately an access token is obtained by providing a user name and password? There are two main benefits of token authentication: (1) If your access token is compromised, then the damage is more limited than if your user name and password were compromised. And (2) an access token can be easily revoked and a new access token generated, whereas revoking and regenerating a user password is more difficult. Accordingly, the idea is that you will request an access token once (or infrequently) and use it going forward, rather than requesting the token anew with every call or every session.

Given an access token, you can now access the Condor Array Telescope data API by supplying the token with every call you make to the API. If your objective is to view (rather than create or update) records, then this will be done exclusively by making HTTP GET requests. This is easily accomplished in python using the "requests" package by specifying your access token in the HTTP headers via

```
1 import requests
2
3 token = '7304045ab7882812aa50c403a17f36c9cc3d1c8f'
4
5 result = requests.get('https://condorarraytelescope.org/api/exposures/',
6   headers={'Authorization': f'token {token}'})
7 print(result.text)
```

This example specifies a request to the /api/exposures endpoint with no additional parameters, which is the most general way of interacting with the Condor Array Telescope database. Details of interacting with the database are described in §§ 4 through 6 below.

4 The /api/exposures/ Endpoint

4.1 General Usage

The "/api/exposures/" endpoint is used to obtain a list of zero of more exposures. Calling the endpoint with no additional parameters returns a (paginated) list of all exposures in the database. For instance, the example code presented at the end of § 3 will print a result similar to

```
1 {"count":60877,
2   "next":"https://condorarraytelescope.org/api/exposures/?limit=5&offset=5",
3   "previous":null,
4   "results":[{"exposure_id":32997,"camera_anti_dew_heater":false,...
```

There are several things to note here:

- The result is paginated, with a default pagination of up to five results per page.
- The result is a dictionary containing four items: (1) a "count" of the total number of results in the request, links to the (2) "next" and (3) "previous" pages, and (4) the "results" in the form of a list.

Each element of the results list contains all of the database fields of one particular exposure, which is uniquely identified by either of the "exposure_id" or "system_filename" fields.

The database fields are organized by the prefix of the field name. In particular, fields prefixed by "camera_," "dust_cover_," "filter_wheel_," "focuser_" "mount_," and "program_" describe properties of the camera, dust cover, filter wheel, focuser, mount, and program, respectively, and fields prefixed by "system_" describe other system properties.

4.2 Supplying Additional Parameters to the Endpoint

Additional parameters may be supplied to the request by using the HTTP GET query-parameter syntax. For instance, you can change the pagination by passing the “limit” parameter, for example as

```
result = requests.get(
  'https://condorarraytelescope.org/api/exposures/?limit=100',
  headers={'Authorization': f'token {token}'}
)
print(result.text)
```

This will provide up to 100 results per page. You can supply multiple parameters by separating the parameters by the“&” symbol. For instance,

```
result = requests.get(
  'https://condorarraytelescope.org/api/exposures/?limit=100&offset=100',
  headers={'Authorization': f'token {token}'}
)
print(result.text)
```

sets the pagination to a limit of 100 with an offset of 100.

4.3 Endpoint Parameters

The following parameters can be supplied to the endpoint:

- array_id: Filter by array_id (currently only 0, since there is only one array).
- element_id: Filter by element_id (currently only 0, since there is only one element).
- file_transfer_status: Filter by file transfer status of the raw data file. Possible values are “pending,” “in process,” and “completed.” Because the raw data are accessible only after they have been successfully transferred, here you probably want “completed.”
- filter: Filter by filter name. Possible values are “g’,” “r’,” “i’,” “L”, “HeII,” “[OIII],” “HeI,” “Halpha,” “[NII],” “[SII],” and “empty.”
- min_datetime: Filter by minimum datetime. Datetime can be specified by any one of Unix timestamp, Unix timestamp in microseconds, Julian date, or UTC-ISO string (which is a string of format, for example, “2021-06-10T12:44:13Z”). (N.B. Due to the well known Y1K bug, any attempt to access data between the assassination of Caesar on the Ides of March, March 15, 44 B.C. and the afternoon of the Battle of Hastings on October 14, 1066 A.D. will cause all raw data to be permanently erased.)
- max_datetime: Filter by maximum datetime, according to the same procedures as for min_datetime.
- ordering: Order by time. Possible values are “newest” and “oldest,” to order by newest first and oldest first, respectively.
- ra, dec, delta_deg: Filter by position on the sky. Here “ra” must be specified by a string of format “HH:MM:SS,” “dec” must be specified by a string of format “+DD:MM:SS,” and “delta_deg” is the search radius in units degree. Unless all three parameters are specified, no filtering by position is performed.

- telescope_id: Filter by telescope_id. Possible values are the integers 0 through 5.
- wcs_status: Filter by WCS of the raw data file. Possible values are “pending,” “in process,” or “completed.” If you require astrometric calibrations, then here you probably want “completed.”

4.4 Raw Database Field Parameters

You can also filter by some (but not all) raw database fields. In particular, you can filter by the following fields:

- camera_frame_type
- filter_wheel_name
- program_comment_1
- program_comment_2
- program_comment_3
- program_observation_type
- program_program_id
- program_program_name
- program_program_user
- program_target_name
- system_exposure_start_timestamp_microsecond
- system_file_transfer_status
- system_preview_transfer_status
- system_telescope_id
- system_wcs_status

In the current version of the API, the raw database field parameters can be filtered only by equality, e.g. as “camera_frame_type=science.”

4.5 URL Encoding

Bear in mind that all parameters passed to the endpoint are transferred via HTTP, so non-Latin characters may need to URL encoded. For example, the value “in process” should be passed to the “file_transfer_status” parameter as “file_tranfer_status=in%20process.” The python urllib.parse library may be helpful here and might be used in a python program like this:

```
import requests
from urllib.parse import quote

filter = quote("g'")

result = requests.get(
  f'https://condorarraytelescope.org/api/exposures/?filter={filter}',
  headers={'Authorization': f'token {token}'}
)
```

5 The /api/exposures/<filename> Endpoint

The "/api/exposures/<filename>" endpoint is used to obtain details of an exposure identified by "filename." So, for example, details of the exposure corresponding to the filename pi-7.2021-06-01-10:41:27.fits.fz might be obtained using

```
import requests

result = requests.get(
  'https://condorarraytelescope.org/api/exposures/pi-7.2021-06-01-10:41:27.fits.fz'
  headers={'Authorization': f'token {token}'})
print(result.text)
```

This returns one element that contains all of the database fields of the exposure corresponding to filename pi-7.2021-06-01-10:41:27.fits.fz.