ALeRCE Python Client

Release 1.0.1

May 11, 2021

Contents

ALeRCE client is a Python library to interact with ALeRCE services and databases.

In this documentation you will find the basic usage of the ALeRCE Client, how to install, a short tutorial for each service and the guidelines to contribute to the project.

CHAPTER 1

Installing ALeRCE Client

The ALeRCE client can be installed through pip with

```
pip install alerce
```

Or clone the repository and install from there

```
git clone https://github.com/alercebroker/alerce_client.git
cd alerce_client
python setup.py install
```

CHAPTER 2

Tutorials

The ALeRCE client is divided according to the requested API, depending on your usecase check the following tutorials:

2.1 ZTF Access

The ALeRCE ZTF API Wrapper gives an easy access to our database through the ALeRCE ZTF API service with Python.

2.1.1 Usage

```
from alerce.core import Alerce
alerce = Alerce()

dataframe = alerce.query_objects()
```

The dataframe will have several columns, the output specification can be found in *Object Response*

2.1.2 Query objects

We will use the *query_objects()* method, this method will query the ALeRCE ZTF API and get a page of results, each result has some statistics of the object and a classification.

The filters are passed as arguments to the method, a list of valid arguments can be found in the *query_objects()* API reference page.

For example, getting all the objects classified as LPV could be done like this:

```
dataframe = alerce.query_objects(
    classifier="lc_classifier",
    class_name="LPV",
    format="pandas"
)
```

You can specify one of the following return formats: *pandas* | *votable* | *json* with json being the default.

Note: If you like to have parameters inside a dict for example that you can reuse later you can do the following:

```
params = {
   "classifier": "stamp_classifier",
   "class_name": "SN",
   "probability": 0.7
}
objects = alerce.query_objects(**params)
```

2.1.3 Getting Classifier List

In ALeRCE we will have multiple classifiers and each classifier will have a list of classes, also called Taxonomy.

For some filters we want to check an specific classifier, to list the current classifiers available we can use the *query_classifiers()* method.

```
# Getting list of classifiers
classifiers = alerce.query_classifiers()
```

The *classifier_name* field should be used as a filter in *query_objects()* method, also if the *classifier_name* and version is already known we can request the list of possible classes with *query_classes()*

```
# Getting classes for a classifier and version
classes = alerce.query_classes("lc_classifier",
                               "hierarchical_random_forest_1.0.0")
```

2.1.4 Querying a known list of objects

You can pass *query_objects* a list of object ids to retreive information of only those objects. You can even apply filters over that list if you wanted to.

```
oids = [
    "ZTF18accqogs",
    "ZTF19aakyhxi",
    "ZTF19abyylzv",
    "ZTF19acyfpno",
]
objects = alerce.query_objects(oid=oids, format="pandas")
```

2.1.5 Query Lightcurve

To get the lightcurve for an object the method *query_lightcurve()* can be used, this will return a dictionary with Detections and Non-detections for that object, also we can get them separetly with *query_detections()*

and *query_non_detections()*.

```
# Getting detections for an object
detections = alerce.query_detections("ZTF18abbuksn",
                                     format="json")

# Getting non detections for an object
non_detections = alerce.query_non_detections("ZTF18abbuksn",
                                             format="json")

# Getting lightcurve for an object
lightcurve = alerce.query_lightcurve("ZTF18abbuksn",
                                     format="json")
```

2.1.6 Query Probabilities

To get the probabilities for an object using the different classifiers implemented by ALeRCE we wil use *query_probabilities()*

```
# Getting detections for an object
probabilities = alerce.query_probabilities("ZTF18abbuksn")
```

2.1.7 Other Queries

There are other more specific queries, to get more information from an object.

To get the features used by the Light Curve Classifier , we can use *query_features()* or *query_feature()* for a single one.

(Check P. Sánchez-Sáez, et al. 2020 for more information on each feature)

```
# Getting all the features for an object as a dataframe
features = alerce.query_features("ZTF18abbuksn", format="pandas")

# Getting multiband period for an object
mb_period = alerce.query_feature("ZTF18abbuksn", "Multiband_period")
```

For each filter ALeRCE calculate some statistics for an object, we can get them with *query_magstats()*

```
# Getting magstats for an object
magstats = alerce.query_magstats("ZTF18abbuksn")
```

2.1.8 Error Handling

The ALeRCE Client has some useful error messages that you can manage when something goes wrong. If you specify a wrong search criteria or no objects were found with your query, then you will get one of the following errors:

- ZTFAPIError (code -1): this is the default error
- ParseError (code 400): this error is raised when there's an error with search parameters
- ObjectNotFoundError (code 404): this error is raised when no objects were returned in your query
- FormatValidationError (code 500): this error is raised when you set a not allowed return format

This errors usually give useful data on what you need to fix with your query. In case you want to do something when an error happens you can capture the error as a regular python exception handling.

```
try:
    data = alerce.query_objects(**my_filters)
except ObjectNotFoundError as e:
    print(e.message)
    # do something else
```

2.2 ZTF Stamps Access

The ALeRCE Stamps API Wrapper gives an easy access to our stamps API that can be used to retrieve stamps and full avro information of a specific alert.

2.2.1 Quickstart

```
from alerce.core import Alerce
#Import ALeRCE Client
client = Alerce()

stamps = client.get_stamps("ZTF18abkifng")
```

2.2.2 Making Queries

There are two operations you can perform with stamps. Getting the stamps of an object and if you are on a jupyter notebook you can plot the stamps.

- *get_stamps()* method will allow you to get stamps of the first detection of an object id. You can also specify a candid to retrieve stamps of a different detection.
- *plot_stamps()* works the same as *get_stamps* but will plot the stamps using IPython HTML if you are in a notebook environment.

Examples

```
# Getting specific stamp
stamps = client.get_stamps("ZTF18abkifng",
                           candid = 576161491515015015)

# Plot stamps on jupyter notebook
client.plot_stamps(oid = "ZTF18abkifng",
                   candid = 576161491515015015)
```

2.3 Crossmatch Access

The ALeRCE catsHTM API Wrapper gives an easy access to our crossmatch API that uses catsHTM.

2.3.1 Conesearch

To get all the objects inside a radius in a specified catalog, we use *catshtm_conesearch()*, where *ra*,'dec' is the center of the search and *radius* is the search radius in [arcsec].

```
from alerce.core import Alerce
#Import ALeRCE Client
client = Alerce()
ra = 10
dec = 20
radius = 1000
catalog_name = "GAIA/DR1"
cone_objects = client.catshtm_conesearch(ra,
                                         dec,
                                         radius,
                                         catalog_name,
                                         format="pandas")
```

Note: Without the *catalog_name* argument the function will return a dictionary with all available catalogs and the value is the conesearch in those catalogs.

2.3.2 Crossmatch

Similar to Conesearch, we look for the objects in a radius, but just get the closest object (*catshtm_crossmatch()*). This method is better used for a small radius.

```
ra = 10
dec = 20
radius = 20
catalog_name = "GAIA/DR1"
xmatch_objects = client.catshtm_crossmatch(ra,
                                           dec,
                                           radius,
                                           catalog_name,
                                           format="pandas")
```

Note: Without the *catalog_name* argument the function will return a dictionary with all available catalogs and the value is the crossmatch in those catalogs.

2.3.3 Redshift

Search if there is available redshift in a *catsHTM* catalog given position and a radius (*catshtm_redshift()*).

```
catalog_name = "GAIA/DR1"
redshift = client.catshtm_redshift(ra,
                                   dec,
                                   radius,
                                   catalog_name)
```

CHAPTER 3

API Reference

3.1 API Reference

class alerce.core.**Alerce**(***kwargs*)

The main client class that has all the methods for accessing the different services.

Parameters

****kwargs** Keyword arguments used for setting the configuration of each service

Attributes

ztf_url The url of the ZTF API

Methods

catshtm_conesearch(ra, dec, radius[, ...])	catsHTM conesearch given an object and catalog_name.
catshtm_crossmatch(ra, dec, radius[, ...])	catsHTM crossmatch given an object and catalog_name.
catshtm_redshift(ra, dec, radius[, format, ...])	Get redshift given an object.
get_stamps(oid[, candid])	Download Stamps for an specific alert.
load_config_from_object(object)	Sets configuration parameters from a dictionary object.
plot_stamps(oid[, candid])	Plot stamp in a notebook given oid.
query_classes(classifier_name, ... [, format])	Gets classes from a specified classifier
query_classifiers([format])	Gets all classifiers and their classes
query_detections(oid[, format, index, sort])	Gets all detections of a given object
query_feature(oid, name[, format])	Gets a single feature of a specified object id
query_features(oid[, format, index, sort])	Gets features of a given object

Continued on next page

Table 1 – continued from previous page

query_lightcurve(oid[, format])	Gets the lightcurve (detections and non_detections) of a given object
query_magstats(oid[, format, index, sort])	Gets magnitude statistics of a given object
query_non_detections(oid[, format, index, sort])	Gets all non detections of a given object
query_object(oid[, format])	Gets a single object by object id
query_objects([format, index, sort])	Gets a list of objects filtered by specified parameters.
query_probabilities(oid[, format, index, sort])	Gets probabilities of a given object

catshtm_catalog_translator	
load_config_from_file	

catshtm_conesearch(*ra*, *dec*, *radius*, *catalog_name='all'*, *format='pandas'*)
catsHTM conesearch given an object and catalog_name.

Parameters

ra [float] Right ascension in Degrees.

def [float] Declination in Degrees.

catalog_name [str] catsHTM Catalog name, *"all"* can be used to query all available catalogs. List of available catalogs can be found in here.

radius [float] Conesearch radius in arcsec.

format [str] Output format [votable|pandas]

Returns

dict Dictionary with the following structure: {

<catalog_name>: astropy.table.Table or pandas.DataFrame or dict

}

catshtm_crossmatch(*ra*, *dec*, *radius*, *catalog_name='all'*, *format='pandas'*)
catsHTM crossmatch given an object and catalog_name.

Parameters

ra [float] Right ascension in Degrees.

def: :py:class:'float' Declination in Degrees.

catalog_name [str] catsHTM Catalog name, *"all"* can be used to query all available catalogs. List of available catalogs can be found in here.

radius [float] Crossmatch radius in arcsec. (Default 100 arcsec)

format [str] Output format [votable|pandas]

Returns

dict Dictionary with the following structure: {

<catalog_name>: astropy.table.Table or pandas.Series

}

catshtm_redshift(*ra*, *dec*, *radius*, *format='votable'*, *verbose=False*)
Get redshift given an object.

Parameters

oid [str] object ID in ALeRCE DBs.

radius [float] catsHTM conesearch radius in arcsec.

format [str] Output format [votable|pandas]

Returns

float Check if redshift is in a catsHTM xmatch response.

get_stamps(*oid*, *candid=None*)
Download Stamps for an specific alert.

Parameters

oid [str] object ID in ALeRCE DBs.

candid [int] Candid of the stamp to be displayed.

Returns

astropy.io.fits.HDUList Science, Template and Difference stamps for an specific alert.

load_config_from_object(*object*)
Sets configuration parameters from a dictionary object.

Parameters

object [dict] The dictionary containing the config.

plot_stamps(*oid*, *candid=None*)
Plot stamp in a notebook given oid. It uses IPython HTML.

Parameters

oid [str] object ID in ALeRCE DBs.

candid [int] Candid of the stamp to be displayed.

Returns

Display the stamps on a jupyter notebook.

query_classes(*classifier_name*, *classifier_version*, *format='json'*)
Gets classes from a specified classifier

Parameters

classifier_name [str] The classifier unique name

classifier_version [str] The classifier's version

query_classifiers(*format='json'*)
Gets all classifiers and their classes

query_detections(*oid*, *format='json'*, *index=None*, *sort=None*)
Gets all detections of a given object

Parameters

oid [str] The object identifier

format [str] Return format. Can be one of 'pandas' | 'votable' | 'json'

index [str] The name of the column to use as index when format is 'pandas'

sort [str] The name of the column to sort when format is 'pandas'

query_feature(*oid*, *name*, *format='json'*)
Gets a single feature of a specified object id

Parameters

oid [str] The object identifier

name [str] The feature's name

format [str] Return format. Can be one of 'pandas' | 'votable' | 'json'

query_features(*oid*, *format='json'*, *index=None*, *sort=None*)
Gets features of a given object

Parameters

oid [str] The object identifier

format [str] Return format. Can be one of 'pandas' | 'votable' | 'json'

query_lightcurve(*oid*, *format='json'*)
Gets the lightcurve (detections and non_detections) of a given object

Parameters

oid [str] The object identifier

format [str] Return format. Can be one of 'pandas' | 'votable' | 'json'

query_magstats(*oid*, *format='json'*, *index=None*, *sort=None*)
Gets magnitude statistics of a given object

Parameters

oid [str] The object identifier

format [str] Return format. Can be one of 'pandas' | 'votable' | 'json'

query_non_detections(*oid*, *format='json'*, *index=None*, *sort=None*)
Gets all non detections of a given object

Parameters

oid [str] The object identifier

format [str] Return format. Can be one of 'pandas' | 'votable' | 'json'

query_object(*oid*, *format='json'*)
Gets a single object by object id

Parameters

oid [str] The object identifier

format [str] Return format. Can be one of 'pandas' | 'votable' | 'json'

query_objects(*format='pandas'*, *index=None*, *sort=None*, ***kwargs*)
Gets a list of objects filtered by specified parameters. It is strongly advised to look at the documentation of **'ALERCE ZTF API'_**

Parameters

format [str] Return format. Can be one of 'pandas' | 'votable' | 'json'

index [str] Name of the column to use as index when format is 'pandas'

sort [str] Name of the column to sort when format is 'pandas'

****kwargs** Keyword arguments. Each argument can be one of the **'ALERCE ZTF API'_** object query parameters.

- **classifier** [str] classifier name
- **class_name** [str] class name
- **ndet** [int[]] Range of detections.
- **probability** [float] Minimum probability.
- **firstmjd** [float[]] First detection date range in mjd.
- **lastmjd** [float[]] Last detection date range in mjd.
- **ra** [float] Ra in degrees for conesearch.
- **dec** [float] Dec in degrees for conesearch.
- **radius** [float] Radius in arcsec for conesearch.
- **page** [int] Page or offset to retrieve. Default value : 1
- **page_size** [int] Number of objects to retrieve in each page. Default value: 10
- **count** [str (bool like)] Whether to count total objects or not. Can be a string representation of boolean like "True", "true", "yes", "false", ...
- **order_by** [str] Column used for ordering. Available values : oid, ndethist, ncovhist, mjdstarthist, mjdendhist, corrected, stellar, ndet, g_r_max, g_r_max_corr, g_r_mean, g_r_mean_corr, meanra, meandec, sigmara, sigmadec, deltamjd, firstmjd, lastmjd, step_id_corr, object, classifier_name, class_name, probability, probabilities
- **order_mode** [str] Ordering could be ascendent or descendent. Available values : ASC, DESC

query_probabilities(*oid*, *format='json'*, *index=None*, *sort=None*)

Gets probabilities of a given object

Parameters

oid [str] The object identifier

format [str] Return format. Can be one of 'pandas' | 'votable' | 'json'

3.2 Models

Here is a list of the response specification for each data model.

3.2.1 Object Response

Table 2: Object Response Model

Name	Type	Description
oid	string	ZTF Object identifier
ndethist	string	Number of spatially-coincident detections falling within 1.5 arcsec going back to beginning of survey; only detections that fell on the same field and readout-channel ID where the input candidate was observed are counted. All raw detections down to a photometric S/N of ~ 3 are included.
ncovhist	integer	Number of times input candidate position fell on any field and readout-channel going back to beginning of survey
mjdstarthist	number	Earliest Modified Julian date of epoch corresponding to ndethist [days]
mjdendhist	number	Latest Modified Julian date of epoch corresponding to ndethist [days]
corrected	boolean	Whether the corrected light curve was computed and can be used
stellar	boolean	Whether the object is a likely stellar-like source
ndet	integer	Total number of detections for the object
g_r_max	number	Difference between the minimum g and r band difference magnitudes (min(g) - min(r))
g_r_max_corr	number	Difference between the minimum g and r band corrected magnitudes (min(corrected(g)) - min(corrected(r)))
g_r_mean	number	Difference between the mean g and r band difference magnitudes (mean(g) - mean(r))
g_r_mean_corr	number	Difference between the mean g and r band corrected magnitudes (mean(corrected(g)) - mean(corrected(r)))
firstmjd	number	First detection's modified julian date
lastmjd	number	Last detection's modified julian date
deltajd	number	Difference between last and first detection date
meanra	number	Mean Right Ascension
meandec	number	Mean Declination
sigmara	number	Right Ascension standard deviation
sigmadec	number	Declination standard deviation
class	string	Highest probability class or according to specified classifier and ranking (Default classifier: *lc_classifier*, ranking: 1)
classifier	string	Classifier name.
probability	number	Highest probability or according to specified classifier and ranking (Default classifier: *lc_classifier*, ranking: 1)
step_id_corr	string	CorrectionStep pipeline version.

3.2.2 Detection Response

Table 3: Detection Response Model

Name	Type	Description
mjd	number	Modified julian Date
candid	string	Alert unique identifier.
fid	integer	Filter ID (1=g; 2=r; 3=i)
pid	integer	Processing ID for image
diffmaglim	number	5-sigma mag limit in difference image based on PSF-fit photometry [mag]
isdiffpos	number	1: candidate is from positive (sci minus ref) subtraction; -1: didate is from negative (ref minus sci) subtraction
nid	integer	Night ID
ra	number	Right Ascension of candidate [deg]
dec	number	Declination of candidate [deg]
magpsf	number	Magnitude from PSF-fit photometry [mag]
sigmapsf	number	1-sigma uncertainty in magap [mag]
magap	number	Aperture mag using 8 pixel diameter aperture [mag]
sigmagap	number	1-sigma uncertainty in magap [mag]
distnr	number	Distance to nearest source in reference image PSF-catalog within 30 arcsec [pixels]
rb	number	RealBogus quality score; range is 0 to 1 where closer to 1 is more reliable
rbversion	string	Version of RealBogus model/classifier used to assign rb quality score
drb	number	RealBogus quality score from Deep-Learning-based classifier; range is 0 to 1 where closer to 1 is more reliable
drbversion	string	Version of Deep-Learning-based classifier model used to assign RealBogus (drb) quality score
magapbig	number	Aperture mag using 18 pixel diameter aperture [mag]
sigmagapbig	number	1-sigma uncertainty in magapbig [mag]
rfid	number	Processing ID for reference image to facilitate archive retrieval
magpsf_corr	number	Corrected PSF magnitude
sigmapsf_corr	number	Error of the corrected PSF magnitude assuming no contribution from an extended component
sigmapsf_corr_ext	number	Error of the corrected PSF magnitude assuming a contribution from an extended component
corrected	boolean	Whether the corrected photometry was applied
dubious	boolean	Whether the corrected photometry should be flagged
parent_candid	number	Parent candid if alert coming from prv_detection (null if no parent)
has_stamp	boolean	Whether the detection has an associated image stamp triplet (when parent_candid = null)
step_id_corr	string	Correction step pipeline version

3.2.3 Non-Detection Response

Table 4: Non-Detection Response Model

Name	Type	Description
mjd	number	Observation Julian date at start of exposure [days]
fid	integer	Filter ID (1=g; 2=r; 3=i)
diffmaglim	number	5-sigma mag limit in difference image based on PSF-fit photometry [mag]

3.2.4 Probability Response

Table 5: Probability Response Model

Name	Type	Description
classifier_name	string	Classifier name
classifier_version	string	Classifier version
class_name	string	Class
probability	number	Probability
ranking	integer	Ranking within the classifier type

3.2.5 Magstats Response

Table 6: Magstats Response Model

Name	Type	Description
fid	integer	Filter ID (1=g; 2=r; 3=i)
stellar	boolean	Whether the object appears to be unresolved in the given band
corrected	boolean	Whether the corrected photometry should be used
ndet	integer	Number of detections in the given band
ndubious	integer	Number of dubious corrections
dmdt_first	number	Initial rise estimate, the maximum slope between the first detection and any previous non-detection
dm_first	number	The magnitude difference used to compute dmdt_first
sigmadm_first	number	The error of the magnitude difference used to compute dmdt_first
dt_first	number	The time difference used to compute dmdt_first
magmean	number	The mean magnitude for the given fid
magmedian	number	The median magnitude for the given fid
magmax	number	The max magnitude for the given fid
magmin	number	The min magnitude for the given fid
magsigma	number	Magnitude standard deviation for the given fid
maglast	number	The last magnitude for the given fid
magfirst	number	The first magnitude for the given fid
magmean_corr	number	The mean corrected magnitude for the given fid
magmedian_corr	number	The median corrected magnitude for the given fid
magmax_corr	number	The max corrected magnitude for the given fid
magmin_corr	number	The min corrected magnitude for the given fid
magsigma_corr	number	Corrected magnitude standard deviation for the given fid
maglast_corr	number	The last corrected magnitude for the given fid
magfirst_corr	number	The first corrected magnitude for the given fid
firstmjd	number	The time of the first detection in the given fid
lastmjd	number	The time of the last detection in the given fid
step_id_corr	string	Correction step pipeline version

CHAPTER 4

Developer and Support documentation

4.1 Need help?

In ALeRCE we work using the Github Repository Issues. To inform a problem check Reporting Issues, and if you need an specific feature check Requesting Features.

4.1.1 Reporting Issues

To report a bug, error or other kind of issue go to Bug Report.

And add a Issue with the *bug* label and the following:

- Issue Description.
- If possible, how to replicate the issue.
- In case of installation or problems with a package the output *pip freeze* to replicate the environment.

4.1.2 Requesting Features

To request a new feature that can be useful for other users/usecases Feature Request with the *enhancement* label.

And adding the following:

- New feature description.
- How it will benefit other usecases.
- If necessary an image with the feature (for example, a feature added to stamp plot in jupyter notebook)

4.1.3 General Question

Create an General Question with the *astronomy* or *programming* label.

4.2 Developer Guide

4.2.1 Configure API for other environment

The ALeRCE client can be modified to request other APIs (for example local or develop APIs). To change the default behavior we have two ways, as initialization parameters or calling a method to change the routes.

As initialization parameters

You can pass parameters to the *Alerce* class constructor to set the parameters for API connection.

```
alerce = Alerce(ZTF_API_URL="https://localhost:5000")
```

From a dictionary object

You can pass parameters to the *Alerce* class from a dictionary object.

```
my_config = {
    "ZTF_API_URL": "https://localhost:5000"
}
alerce = Alerce()
alerce.load_config_from_object(my_config)
```

Routes that can be modified

The examples changes the default ZTF api, but there are other API and routes that can be modified, all the routes are the following

Table 1: Routes

Variable	Default	Description
ZTF_API_URL	http://api.alerce.online/ztf/v1	ALeRCE ZTF API route.
ZTF_ROUTES	{ 'objects':'/objects', 'single_object':'/objects/%s', 'detections':'/objects/%s/detections', 'non_detections':'/objects/%s/non_detections', 'lightcurve':'/objects/%s/lightcurve', 'magstats':'/objects/%s/magstats', 'probabilities': '/objects/%s/probabilities' }	Dictionary with query type and route, with %s as wildcard for object id
CATSHTM_API_URL	http://catshtm.alerce.online	ALeRCE catsHTM API base url.
CATSHTM_ROUTES	{ "conesearch": "/conesearch", "conesearch_all": "/conesearch_all", "crossmatch": "/crossmatch", "crossmatch_all": "/crossmatch_all" }	ALeRCE catsHTM routes.
AVRO_URL	http://avro.alerce.online	ZTF AVRO/Stamps API
AVRO_ROUTES	{ "get_stamp": "/get_stamp", }	ZTF AVRO/Stamps API Routes

4.2.2 How to contribute

We are open to contributions in new features or fixing issues.

To send a contribution add the #IssueNumber in the Pull Request (PR) for Issue tracker, the PR then will be assigned to the team to be reviewed.

Steps to create a Pull Request

1. Fork the repository
2. Create a branch if necessary
3. Fix the issue or add new feature
4. Push changed to personal repository
5. *Create a PR <https://github.com/alercebroker/alerce_client/pulls>* to the *alercebroker* repository

For a detailed guide check this link

Index