# D3.3.3: TiDES Target selection from broker

## WP3.3 Spectroscopic classification of transients

**Project Acronym** LUSC-B

**Project Title** UK Involvement in the Legacy Survey of Space and Time

**Document Number** LUSC-B-29

| | |
|---|---|
| **Submission date** | 13/MAY/2022 |
| **Version** | 1.0 |
| **Status** | Final |
| **Author(s) inc. institutional affiliation** | Christopher Frohmaier (Southampton)<br>Mark Sullivan (Southampton) |
| **Reviewer(s)** | Stephen Smartt (QUB)<br>Matt Nicholl (Birmingham) |

| **Dissemination level** | |
|---|---|
| Public | |

# Version History

| Version | Date | Comments, Changes, Status | Authors, Contributors, Reviewers |
|---|---|---|---|
| 0.1 | 13/APR/22 | First draft | |
| 0.2 | 13/MAY/22 | Response to referee comments | |
| 1.0 | 20/JUN/22 | Exec Group approved | |
| | | | |
| | | | |
| | | | |

# Table of Contents



# List of Figures



# List of Tables

# 1 Executive Summary

The Legacy Survey of Space and Time (LSST) and the 4m multi-object spectroscopic telescope (4MOST) commence survey operations in late 2023 and early 2024 respectively. The Time-Domain Extragalactic Survey (TiDES) on 4MOST will follow-up LSST discovered transients to obtain spectroscopic measurements for tens-of-thousands of supernovae, galaxies, active-galactic nuclei (AGN), and strongly-lensed systems. For the transients and their hosts, this data will allow us to map the astrophysical diversity of cosmic explosions and measure the equation of state parameter for dark energy to unprecedented precision. TiDES forms the basis of WP3.3: Spectroscopic classification of transients.

A key aspect of performing a reliable and efficient follow-up of transients is to create a discovery infrastructure within the LSST alerts stream. Access to this stream is served by several community brokers, allowing users to create custom filters and use value-added data products to fine-tune a selection algorithm to flag transients suitable for their science programme. This is exactly the use-case for TiDES and forms the core of this deliverable.

Given that the start of LSST survey operations is a couple of years away, we are developing our selection algorithms on the Lasair-processed ZTF alerts stream. However, we have had to change the scope of our deliverable as ZTF suffered a prolonged outage during a key phase of the development cycle. As a result, this deliverable will not include the Lasair Kafka stream of ZTF alerts or our real-time Lasair filter to pre-select candidates. Instead, we present software that applies a customisable light curve-based selection function on any ZTF object from the archive of historical observations. This mimics a real-time feed, stepping through the observations and assessing each object against the user-defined selection criteria. The planned integration of this with a real-time feed will instead be presented in a future deliverable and the further down-stream connections with 4MOST will be implemented with deliverable D3.3.4.

# 2 Introduction

The software presented in this deliverable is intended to evaluate transient light curves to check whether they meet a specific set of criteria. Work Package 3.3 focuses on the spectroscopic classification of transient events from LSST using the 4MOST/TiDES facility. The majority of extra-galactic transients discovered by LSST being supernovae (SNe), and with Type Ia SNe playing the critical role in our cosmology analysis. 4MOST/TiDES will survey the entire southern sky, collecting spectra of transients and their hosts at an industrial scale, unrivalled in volume by an contemporary facility. TiDES will not, however, dictate the overall strategy or any individual pointing of 4MOST during its operations, nor will TiDES know precisely where 4MOST will observe on any given night. This means TiDES will need to be ready with a target list to accommodate 4MOST observations where ever and whenever they occur. It is, therefore, paramount that TiDES have the software capabilities to rapidly identify targets from the LSST real-time stream. Given that LSST is still under-construction, we are using the Zwicky Transient Facility (ZTF) as our development survey with the intent to scale-up operations in the coming years. In this document we present a small piece of software that can evaluate a ZTF light curve and indicate to the user if the target passes or fails a custom light curve-based selection requirement requirement. All the code presented in this document are available from the following GitHub repository: https://github.com/lsst-uk/tidesInterface-WP3.3

## 2.1 Change history of the deliverable

This deliverable is a key element of the October 2021 to March 2022 cycle and was originally envisaged to incorporate a real-time analysis of the ZTF stream from the Lasair broker. We began development and testing of a custom filter on the Lasair system, followed by early work on implementing a Kafka stream. In late 2021, however, the cryo-cooler on the ZTF camera suffered an issue that required down-time to repair. After this was fixed, another issue was reported in early 2022 relating to zeolite particle contamination of the instrument which needed additional cleaning. Due to the prolonged period without real-time data, D3.3.3 was changed to remove a dependence on the real-time alert stream. Instead we focus on assessing historical light curves against our ability to create custom selection criteria.

## 2.2 Glossary of Acronyms

| | |
|---|---|
| **4MOST** | **4**m **M**ulti-**O**bject **S**pectroscopic **T**elescope |
| **DESC** | **D**ark **E**nergy**S**cience **C**ollaboration |
| **LSST** | **L**egacy **S**urvey of **S**pace and **T**ime |
| **OB** | **O**bserving **B**lock |
| **TiDES** | **Ti**me-**D**omain **E**xtragalactic **S**urvey |
| **ZTF** | **Z**wicky **T**ransient **F**acility |

# 3 Requirements

The entirety of this code is written in the `Python` language, v3.8 was used, but all versions of Python 3 should be compatible. Standard inbuilt Python libraries are needed, with additional libraries including: `numpy`, `pandas`, `json`, `YAML`, `docopts` and `matplotlib`. These required libraries are listed in the `requirements.txt` file. The Lasair Python library (`lasair`) is used to interface with the Lasair broker to retrieve ZTF light curves. This library is still in the beta phase of development – v0.0.3 at the time of writing. We expect evolution in the package and will maintain compatibility to the official and stable release. Furthermore, the Lasair API is currently only returning complete data for objects observed after 2020, this somewhat limits our ability to test the code against the full ZTF history, but it is adequate to demonstrate the deliverable. The Lasair Python package can be installed following the instructions here: https://pypi.org/project/lasair/

## 3.1 API Tokens

Accessing Lasair services through either the Python library or API requires an authorisation token. For the purposes of this deliverable, the access token can be read from a YAML file or directly used as a command line argument. An example of the contents of a YAML access file is shown below, however this script only requires the `token` keyword. For security reasons, files containing tokens should be kept away from main development or execution areas. The Lasair API token must be separately obtained from your username and password following the instruction here: https://lasair-iris.roe.ac.uk/api

```
lasair:
    username: LHamilton44
    password: 7xWDC0820
    token: Cb6442e76373g2M7bwjd738ae1c946d54b8f7993
```

# 4 Executing the script

In this Section we detail the process of executing the code in this deliverable. The directory structure, starting from the `./tidesInterface-WP3.3` directory of the Github repo (https://github.com/lsst-uk/tidesInterface-WP3.3) is as follows:

```
./tidesInterface-WP3.3
+-- LICENSE.txt
+-- README.md
+-- Software Management Plan for TiDES WP3.docx
\-- tidesTargeting
    +-- checkObjects.py
    +-- requirements.txt
    +-- developmentNotebook.ipynb
    +-- tidesSelectionFunctions.yml
    \-- ztfIaList.dat
```

Future deliverables will also be developed within this repo, but for D3.3.3 we will work within the `tidesInterface-WP3.3` sub-directory. We begin by showing the main execution of the Python script `checkObjects.py`, followed by an explanation of the input arguments.

```
python checkObjects.py
```

```
-k /path/to/lasair/token/yaml/file.yml (or just the token)
-s /path/to/selection/function/file.yml
-n nameOfSelectionFunction
-i /path/to/ztf/objects/file.csv
-o /path/to/dave/output/
-p Plots[[True]/False]
-c 50 [default: 50]
```

- `-k`: This argument holds the full path to the YAML access file containing your Lasair token described in Section 3.1. Alternatively it can take your Lasair token directly as input.
- `-s`: This is another path to a YAML file, but this one contains the selection function criteria described in Section 4.1. Note: Both the `-k` and `-s` arguments can point to the same file as long as the relevant information is held within.
- `-n`: Within the selection function file, the specific criteria are held under sub-categorised names. This input takes the name of the criteria. See Section 4.1 for details.
- `-i`: This argument should point to an input file containing the ZTF names of all objects you wish to evaluate. See 4.2 for guidance.
- `-o`: Point this argument to a folder you wish to save the output files to. If the directory does not exist, it will be created for you.
- `-p`: A simple binary input of True/False that asks the user if they want light curves plotted for each object analysed and saved in the `-o` directory.
- `-c`: The maximum number of objects queried per API call. This value defaults to 50 and is limited to 50, if a number greater than this is entered (or no number at all), then it supplies the default.

## 4.1 The Selection Criteria

The light curve-based selection criteria are stored in YAML files. Each key in the file defines a different set of requirements that a light curve must meet. If additional parameters are desired, then the Python script must also be adjusted accordingly. Below, we show the example contents of such a file. At a minimum, the following parameters need to be defined:

- `filters`: The filters for consideration. For ZTF public data, only `g` and `r` data are available.
- `significance`: This defines the signal-to-noise threshold an observation must pass. (SNR $\geq$ `significance`) .
- `minBands`: The minimum number of separate bands in the `filters` list that must meet the `significance` criterion.
- `minNights`: The minimum number of separate nights that qualifying light curves points must appear across.
- `magLimit`: At least one light curve point (in any band) needs to be brighter than this limit in magnitude-space. For TiDES supernovae we have set this as the approximate 4MOST 1-hour exposure depth.

```
tidesSNZTFSelect:
    filters: ['g', 'r', 'i', 'z']
    significance: 5
    minBands: 2
    minNights: 2
    magLimit: 22.5

tidesSNLSSTSelect:
    filters: ['g', 'r', 'i', 'z']
    significance: 5
    minBands: 3
    minNights: 2
    magLimit: 22.5

tidesHostLSSTSelect:
    filters: ['g', 'r', 'i', 'z']
    significance: 5
    minBands: 3
    minNights: 2
    magLimit: 23
```

In the command line execution of the script, the `-n` flag would be followed by a key, such as `tidesSNZTFSelect` or similar.

## 4.2 Input list of targets

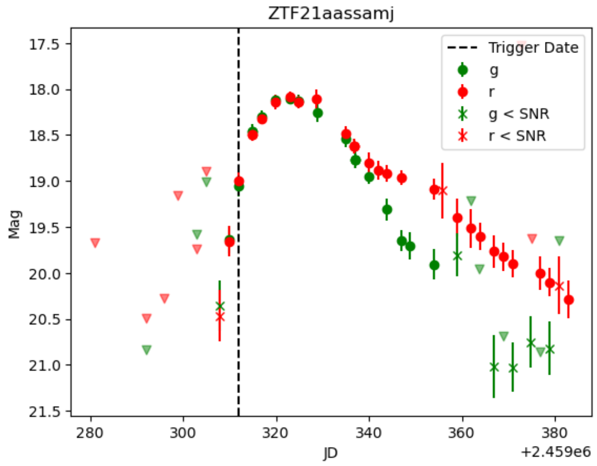
The input file for this deliverable acts as a placeholder for the original intention of interfacing with the Lasair real-time alerts stream. Given that we have paused our development of this, we have chosen to accept instead a list of ZTF objects for individual evaluation. The target catalogue input is an ascii file with a ZTF object name on each line. Each object is evaluated individually by the code. An example file of ZTF objects (`ztfIaList.dat`) obtained from the ZTF Bright Transient Survey is contained in the Github repo. The full path to any file containing ZTF object names would follow the `-i` flag.

## 4.3 Expected Output

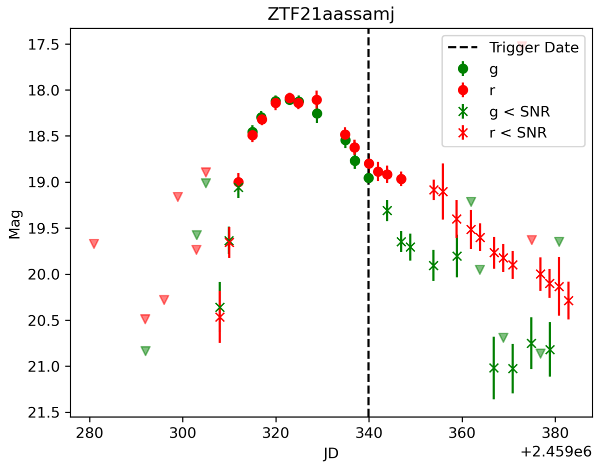
The script works by reading the input list of ZTF targets and splitting the file into groups with size specified by the `-c` parameter. The Lasair API is then queried to return the current light curves for all objects in that group. Each object is analysed sequentially. Given the data volumes from ZTF in early testing of the real-time stream, the processing time was reasonable on a single user's laptop without any multi-processing enhancements. However, if the resource requirements for the LSST stream prove to be too demanding, then a small upgrade to the code would be to process each group in parallel. At the end of the execution, each group is concatenated back into a single list with additional data results and returned to the user.

To analyse each light curve, and best replicate a real-time stream, the observations are analysed chronologically with each subsequent observation prompting a check against the observing criteria specified in the selection function. The first epoch that satisfies all the criteria becomes the spectroscopic triggering date. During the actual operations of 4MOST/TiDES, this 'trigger' will result in the creation of an OB that is delivered to 4MOST for spectroscopic follow up observations. Development on this plan is schedule for the next cycle, with completion as part of D3.3.4.

Figures are created during run-time of the pipeline by default, but can be toggled using the `-p` flag. The effect of selection criteria can be seen in Figure 1. A less conservative selection




(a) An observing criteria of $>5\sigma$ detections, on 2 separate nights, in 2 different bands is considered for this object.




(b) An observing criteria of $>10\sigma$ detections, on 10 separate nights, in 2 different bands is considered for this object.

Figure 1: A comparison of the object ZTF21aassamj under two different selection functions. Panel (b) uses a much more conservative requirement to demonstrate the effect of changing selection functions to the date of spectroscopic triggering.

has been applied to the light curve in Figure 1a compared to that in Figure 1b. The most direct and obvious result of changing the selection criteria is to the spectroscopic triggering date. During LSST alert stream tests, commissioning, or early-science, TiDES should investigate the most suitable selection criteria on light curves from the real-time stream. We must achieve not only a high-purity of real transients in the 4MOST spectroscopic follow up queue, but also identify these objects early enough to maximise the visibility time and hence the probability that 4MOST will obtain a spectrum. This is where the strength of the YAML format comes into play; we will be able to map an entire selection parameter-space and test each in parallel against the incoming transients. This will allow us to find the sweet-spot for TiDES which we can then lock-in for the remainder of the survey. Furthermore, custom selection functions can be created in the event TiDES wanted to employ multiple strategies for selection. For example, we may wish to have a selection function to identify objects for a cosmology analysis and another, more speculative, function for young transients with less significant early detections. This would require the 'tagging' of individual objects to see which pipeline identified them so that proper bias modelling can be performed at a later date. Additional criteria in the YAML file would need some changing of the code presented in this deliverable (such as light curve color) but would be achievable with only small effort in the future.

The final output from this code is a simple text file, stored in the specified output directory with the filename `PassFailCut.csv`. Within the broader picture of the TiDES pipeline, this will not be the final output. Rather it is simply a placeholder until we develop a communication stream with 4MOST. The csv file contains three columns: `ZTFName, PassCut, TriggerDate`. The first column is self-explanatory and taken from the input file, the second column will give a True/False return based on whether the object met the selection criteria or not. In the event the Lasair API does not return any light curve information, 'No data' will be output. The third column contains the Julian Date when the object met the selection criteria or -9999 if it does not. A small example of the output file contents is shown below.

```
ZTFName,PassCut,TriggerDate
ZTF20aayeims,No Data,-9999.0
ZTF20aayjxdv,No Data,-9999.0
ZTF21acmresp,True,2459527.0531481
```

```
ZTF21acngvxd,True,2459531.9840741
ZTF21acnveyk,True,2459528.9929977
ZTF21acojncf,False,-9999.0
ZTF21acoqbfm,True,2459530.9920255
ZTF21acoqbuo,True,2459530.9423843
ZTF21acouzjv,True,2459532.9143866
ZTF22aaafomy,True,2459625.9417593
```

## 5 Future Integration and development within the TiDES Pipeline

This deliverable represents an intermediate point of the whole WP3.3 pipeline development. We have demonstrated a method of adjusting selection criteria parameters for ZTF transients with the broader goal of adapting this to LSST light curves. We still need to implement communication with real-time data streams from the community brokers. While this deliverable integrates closely with the Lasair broker, we must remain open to collaboration with future LSST-focused brokers, e.g. the DESC broker. This will require some adaptation of our deliverable but will not result in a change of the fundamental concepts shown. At the time of submission, the ZTF real-time stream had returned, we will resume work on our real-time analysis in the next project cycle. The longer term goal of our Phase-B development also demands communication with the 4MOST targeting server. This will require us to adapt the output of D3.3.3 to create OBs compatible with the 4MOST requirements. These OBs will have relatively short expiration dates, but will be refreshed and updated each time a new observation is obtained for any given transient. Ultimately, the output and data storage will also need to be adapted to move away from our placeholder csv files, most likely we will move to a relational database structure of continually updated TiDES targets to bridge the LSST and 4MOST systems.