lasair Documentation

Release v4.1.0

Lasair Team

2023

START

Lasair is being developed by The University of Edinburgh, Queen's University Belfast, and Oxford University with the ultimate goal of serving transient alerts from the future Rubin Legacy Survey of Space and Time (LSST) to the astronomical community. *Lasair* (pronounced '*L-AH-s-uh-r*') means flame or flash in Scots and Irish gaelic. To prototype functionality needed to process transient event alerts from LSST, Lasair is currently processing and serving data from the public stream of the Zwicky Transient Facility (ZTF), which is releasing a transient alert stream in a format similar to that envisaged for LSST. We thank ZTF for access to this valuable public data stream.

If you make use of Lasair in any of your work, please remember to cite our paper: Lasair: The Transient Alert Broker for LSST:UK, K. W. Smith, R. D. Williams et. al., Research Notes AAS, **3**,26 (2019).

To start using Lasair to filter the deluge of alerts:

CHAPTER

ONE

QUICK START

Lasair is built as a platform to enable scientific discoveries from the dynamic

Universe. Its input is a transient sky survey such as ZTF or LSST, that find changes in

brightness in the night sky, each called an "alert". Such alerts may result

from supernovae, active galaxies, merging neutron stars, variable stars, and many other

astrophysical phenomena (see here for more).

Alerts from the same place in the sky are combined to objects.

The alerts provide the brightness of the object with time –

see here for more.

Lasair adds information to the object, matching the position with the known

astronomical catalogs – see here.

The transient surveys provide large numbers of alerts: about 400,000 per night

from ZTF when the sky is clear at Palomar in California, rising to millions per

night when the Rubin Observatory in Chile

is running its flagship LSST survey. Far too many for a human to consider!

Therefore the primary duty of a broker like Lasair is to *filter* the stream to

concentrate what is wanted and discard that which is not. In this section

we show **how to make a Lasair filter**,

specifically the one used for building the set of alerts shown on the Lasair front page. That display is made from recent, bright,

real alerts that are identified with known classes of stars and galaxies.

If you click on any of the red, orange, blue, or yellor markers, you will see

a popup witha link to the full object page, the age of the most recent alert,

its magnitude, and its class.

Each object in the Lasair objects table has a lot of columns in several tables,

and for this example we will concentrate on just a few:

- From the objects table:
  - `objectId`: The identifier for an object that is used to link to the full
    object page,

   - `ramean, decmean`: The position of the object in the sky, to place it

     correctly,
   - `gmag, rmag`: the magnitudes of the latest alert in the g and r filters,
   - `jdmax`: the Julian Day (i.e.date and time) of the latest alert,
   - `jdnow()`: an SQL function that returns the Julian Day now, so we can

     subtract to get the age in days,
   - `ncandgp`: number of good, positive alerts belonging to this object.
- From the sherlock_classifications table:
   - `classification`: Sherlock class according to the sky context – see core_functions/sherlock.html for more.

1.1 Create New Filter

We can build the filter by clicking on 'Filters' in the Lasair sidebar, then

the red button 'Create New' at top right.

For your first filter, you won't be using any of the dropdowns for Watchlist,

Watchmap, or Object Annotators, you'll fill in the black textarea labelled **SELECT COLUMNS** and **WHERE**.

Type these lines in the SELECT COLUMNS. Each line is explained in the dropdown.

`objects.objectId,`

Notice that as you type, the intelligent autocomplete makes suggestions. Don't forget the comma at the end.

`objects.ramean, objects.decmean,`

The word *mean* is because this is the average position of the multiple alerts that are part of the same object. Don't forget the comma at the end.

`objects.gmag, objects.rmag,`

The g or r magnitude for the most recent alert. Each alert is done with one of the filters, so either *gmag* or *rmag* will be *NULL*.

`jdnow()-objects.jdmax AS age,`

This SQL fragment subtracts the Julian Day now from the Julia Day of the alert, and renames the result as *age*.

```
sherlock_classifications.classification AS class
```

This attribute is from a different table, the Sherlock classification of the object. The long name is renamed as the much simpler *class*.

You see as you type that the tables you are using appear in the middle of the

three black textareas,

labelled **FROM**.

Now type these lines into the **WHERE** box:

```
objects.jdmax > jdnow() - 7
```

We select only those objects whose most recent alert has been in the last 7 days.

```
AND (objects.gmag < 17 OR objects.rmag < 17)
```

We want bright objects only, mostly to cut the numbers being drawn on the Lasair front page. Give that one of the attributes is *NULL* the *OR* selects the one that is not, and requires it to be less than 17. Don't forget the *AND* at the beginning.

```
AND objects.ncandgp > 1
```

There are a lot of 'orphans' in the Lasair database, that have only one alert. Many of these are not worth looking at, so we require the number of candidates to be greater than 1.

```
AND sherlock_classifications.classification in ("SN", "NT", "CV", "AGN")
```

These codes are for the different Sherlock classifications: possible supernova, nuclear transient cataclysmic variable, active galaxy.

1.2 Run your filter

You can simply run the filter on the existing database by clicking the red

button 'Run Filter'.

You should see a table of the recent alerts, the same set as are on the Lasair

front page.

You can click on the column headers to sort, and click on the `objectId` to go

to the detail

for any of the objects.

1.3 Save your filter

But doing more with Lasair requires an account – its just a simple matter of

entering

your valid email address – see here to register.

Click the black button 'Save' on the create fulter page, then fill in the

details: Name and Description, and you can choose to make it public, so that it

appears in the [Public Gallery]((https://lasair-dev.lsst.ac.uk/filters). Once its shared like

this, others can use it, or copy and modify it. Another option in the Save

dialogue

has three choices:

* muted: The filter is saved, and you can run it and edit it

* email stream (daily): Means that you receive an email – at the address of

your Lasair account –

whenever an alert causes an object to pass through the filter.

This is restricted to one email in 24 hours.

* kafka stream: The substream induced by the filter becomes a kafka stream –

see here for more.

Other options on the filter page bring in other tablesin addition to teh `objects` table

– see the schema browser for the full list. These

include:

- `sherlock_classifications`: the results of an intelligent matching of

  multiple catalogues

  with the position of the alert on the sky – see here for more.

- `crossmatch_tns`: you can filter your results to be alerts coincident with the TNS name server. You can select supernova types ,

  dscovery date, and more.

- `watchlist`: you can filter your results to be only those coincident with a

  list of sources that you or someone else has uploaded – see here for more.

- `watchmap`: you can filter your results to be only those inside a sky area

  that you or someone else has uploaded – see here for more.

- `annotation`: you can find events that have been classified or otherwise

  annotated external to Lasair. You can also set up your own annotation service – see here.

The following is about the astronomical science enabled by Lasair:

CHAPTER TWO

ABOUT LASAIR

The Rubin Observatory
will provide unprecedented temporal resolution, depth and
uniform photometry over an entire hemisphere, along with a real-time stream of
alerts from the ever changing sky. To extract the scientific potential from
that stream, the community needs brokers that offer the ability to filter,
query, and manipulate the alerts, and combine them with external data sources.
The LSST:UK consortium
has been building just such a broker Lasair, alongside
an International Data Access Centre (IDAC), building on its strengths and
heritage in leading astronomical surveys, data processing and analysis. The hope
is that Lasair will be of value to the worldwide community, not just to the the UK consortium.

CHAPTER

THREE

THE LASAIR APPROACH

the science itself.

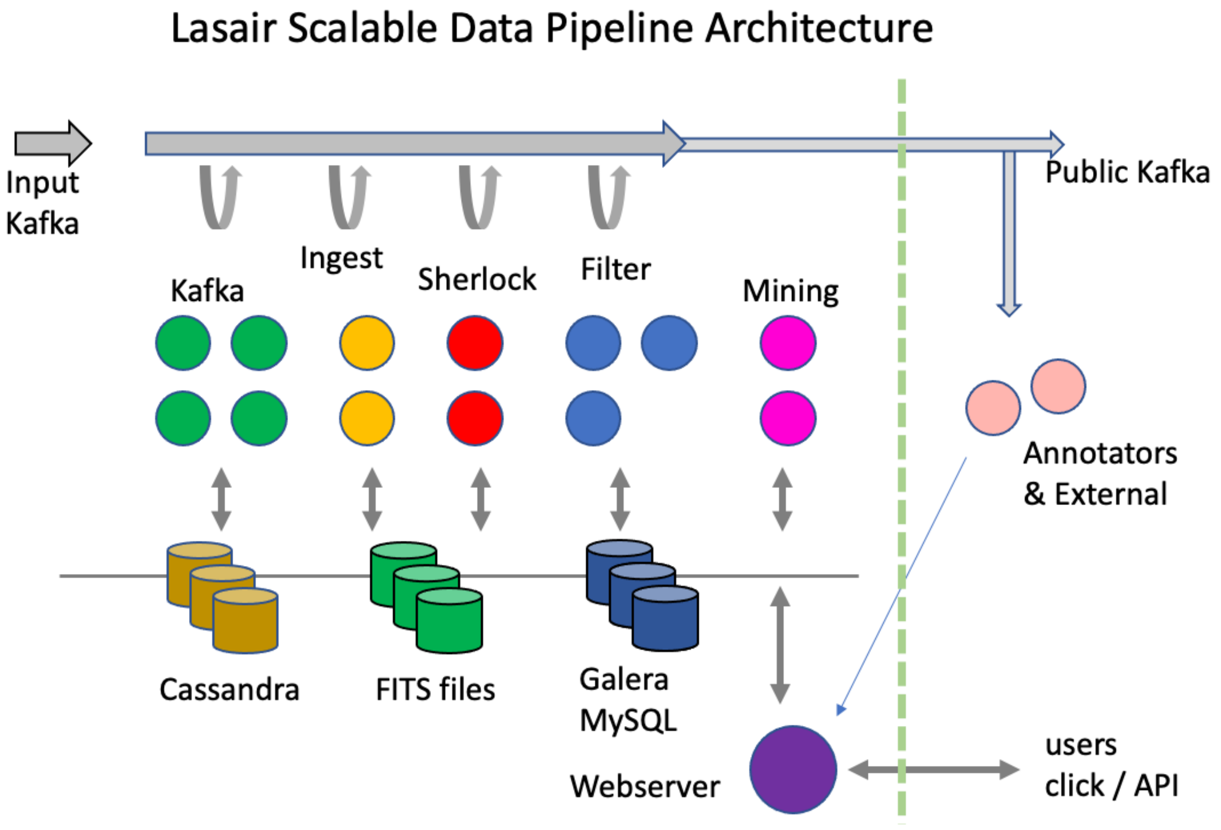
Every LSST broker aims to filter the stream, but Lasair does this differently.
Rather than scientists making python code that needs to be vetted, Lasair
offers direct access with a staged approach: scientists can start with a
simple, immediate mechanism using familiar SQL-like languages. These SQL-like
queries can be custom made or users can choose and modify one of our pre-built
and tested queries. These queries return an initial selection of objects, based
on our rich value-added data content, and users can then run their own local
code on the results. Users can build up to running their own code on both the
stream and the database with high-throughput resources in the
UK science collaboration called IRIS. The
SQL filters and code can be made public, shared with a group of colleagues,
copied, and edited.
SQL filters can be escalated from static (run on command) to streaming filters,
that run whenever new alerts arrive. A broad overview of the Lasair design is
given in Figure 1.

CHAPTER

FOUR

HOW LASAIR WORKS

Lasair runs in the Somerville computing cloud at the Advanced Computing Facility near Edinburgh, Scotland.



Lasair ingests data with a pipeline of clusters: each cluster does a different job, some more compute/data intensive than others, so it is difficult to know a priori how much resource should be allocated to each. Our design gives flexibility: each cluster can be grown or reduced according to need. Also, there are various persistent data stores, again, each is driven by a resilient cluster that can be grown or reduced according to need. Figure 1 shows the concept: data enters the Kafka system on the left and progresses to the right.

The green cluster reads, processes, and puts different data into the Kafka bus; as soon as that starts the yellow cluster pulls and pushes; eventually the whole pipeline is working. The clusters may also be reading and writing the data stores.

We also include the web and annotator nodes in this picture (bottom and right), as well as the mining nodes, although they are not part of the data ingestion pipeline. The web server nodes support users by delivering web pages and responding to API requests. The annotator nodes may be far from the Lasair computing centre and not controlled by us, but they are in this picture because just like the others, they push data into the data storage and may read from Kafka.

The Kafka system is represented by the green nodes in Figure 2 as well as the grey arrow at the top. It is responsible for reading and caching the alert packets from the USA, as well as sending it to the compute nodes and receiving their resulting packets.

The Ingest nodes read the original alerts alerts from the Kafka system, and puts the cutout images in the shared filesystem, the recent lightcurve to NoSQL (Cassandra) database, then reformats the alert as JSON – since there is no binary content – then pushes that into the Kafka system.

Each Sherlock node has a SQL database of 5 Tbytes of astronomical sources from ~40 catalogues. The sky position of the input alert is used to intelligently decide on the most likely associated source from the catalogues, finding out, for example, if the alert is associated with a known galaxy, or if the alert is a flare from a known CV (cataclysmic variable).

Each filter node computes features of the 30-day light curve that comes with the alert (a year with LSST), as well as matching the alert against user-made watchlists and areas. Records are writen to a local SQL database onboard the node for the object and features, the Sherlock data, the watchlist and area tags. Other tables have already been copied into the local database from the main SQL database (see Background Services below). After a batch of perhaps 10,000 alerts are ingested to the local database, it can now execute the user-made queries and push out results via the public Kafka system – or via email if the user has chosen this option. The tables in the local database are then pushed to the main SQL database and replace any earlier information where and object is already known. Once a batch is finished, the local database tables are truncated and a new batch started.

The Lasair webserver and API server allow users to control their interactions
with the alert database and the stream. They can create a watchlist of their
interesting sources, and Lasair will report crossmatches with members of the
watchlist. They can define regions of the sky, and Lasair will report when
alerts fall inside a region. They can define and save SQL queries that can run
in real time as filters on the alert stream.
The Lasair API supports annotation: a structured external packet of extra
information about a given object, that is stored in the annotations table in
the SQL database. This could be, the result of running a machine-learning
algorithm on the lightcurve, the classification created by another broker, or
data from a follow-up observation on the object, for example a link to a
spectrum. Users that put annotations into the Lasair database are vetted, and
administrators then make it possible. That user will run a method in the Lasair
API that pushes the annotation: all this can be automated, meaning the
annotation may arrive within minutes of the observation that triggers it.

CHAPTER FIVE

ZTF AND LSST

The Lasair project splits into two: the existing working version, Lasair-ZTF, that has been ingesting and exposing alerts from the ZTF survey for two years; and the future version Lasair-LSST, which is being developed based on the lessons learned from Lasair-ZTF. We are keeping the essentials of the user interface of Lasair-ZTF (static and streaming SQL queries, full database access, watchlists, classification and annotation), but are rebuilding the backend architecture for LSST event rates, using parallel services and scalable software.

In the timeframe beyond the first data releases of LSST, we can expect continuing change. New surveys will come on line, new robotic follow-up systems, and new classification systems will proliferate, leading to more real-time transient streams and derived information. Just as cross-matching of static catalogues has gained importance in the last years, so *dynamic* cross-matching will become an engine of discovery: those transients observed by A and B, or classified in contradiction by C and D. While the bulk of data will continue to be dominated by Rubin data, there will be a deluge of metadata and annotation. Lasair will be well-suited to this challenge, building on our existing mechanisms for dynamic cross-match (e.g. the IAU's Transient Name Server) and utilising our flexible schema system. Lasair will add new tables and schemas to our databases, and build information systems to make it easy for scientists to navigate the deluge of metadata.

CHAPTER

SIX

SCIENTIFIC GOALS OF LASAIR

We aim to facilitate all four science themes of LSST within the Lasair platform: Dark Matter and Dark Energy, the Solar System, the Changing Sky, and the Milky Way. We will do this by providing combined access to the alerts, to the annual data releases, and to external data sources, and by providing a flexible platform which creative users can adapt to their own ends. Design of Lasair is driven by a detailed Science Requirements Document which is available on request. We will have a review with broader international input if we are selected. Below we explore the issues arising from key science topics.

6.1 Extragalactic Transients

Luminous transients outside our own galaxy include supernovae, kilonovae, tidal disruption events and AGN flare activity, nuclear transients of unknown origin, gamma-ray bursts, stellar mergers, white dwarf - NS/BH mergers, super-luminous supernovae and fast blue optical transients.
These have timescales from hours (GRB afterglows), days (kilonovae and WD-NS/BH mergers), to weeks (supernovae, fast blue optical transients), months (TDEs, SLSNe, AGN activity) to years (SLSNe at high redshift, AGN and nuclear transients, SNe from CSM interaction).
All of this science requires lightcurves, links to galaxy and redshift catalogues, precise astrometric cross-matching, correlation with high energy information, multiwavelength cross-matching and our concept of ``tagging'‘ which we introduce here. Objects need to be found on timescales of minutes to years due to the intrinsic timescale, which is mostly driven by the mass ejected by the transients (through photon diffusion time). Some scientific highlights that Lasair will enable are :

6.2 Kilonovae and gravitational wave sources

Users will be able to select their own candidates by combining colour,
lightcurve evolution, host galaxy information and any multi-wavelength
coincidences using SQL, kafka filtering, or the Lasair API (see section 4).
This can be used to enable searches for all ``fast-transients'‘ of timescales
of minutes to days (e.g. GRB afterglows, orphan afterglows, WD-NS/BH mergers).

6.3 Massive samples of supernovae

Lasair will link all transients to a list of likely host galaxies together with
their photometric redshifts and their spectroscopic redshifts, should they
exist. We are working closely with the two major ESO projects that will provide
tens of thousands of spectra for LSST supernovae.
We will coordinate SN discoveries in Lasair with spectra from the 4MOST
multi-fibre spectrometer on the ESO VISTA telescope.
We will provide DESC with the ability to select 35,000 live transients for
spectra and obtain spectra of 70,000 host galaxies in the TiDES (Time Domain
Extragalactic Survey).
This will provide the largest cosmological sample of type Ia SNe, together
with a massive statistical sample to understand supernova explosion physics
across a range of redshifts and host galaxy masses and metallicities. Lasair
will provide both (reproducible) selection and extract the scientific content
(type, phase, redshift etc) to re-ingest into the broker for user exploitation.
We are also working closely with the UK team responsible for the science
software infrastructure behind SOXS on ESO's New Technology Telescope. This is
a 0.35-2μm spectrometer and ESO are fully dedicating the NTT to time domain
science, with the schedule being run by the SOXS consortium.
We will enable the SOXS marshall and rapid data analysis pipeline to interface
with Lasair, to select LSST transients ($\sim$ few$\times 10^3$) for
classification and re-ingest the information and public data for all users to
access.

6.4 AGN, TDEs and long lived transients

Similar to the above, we will allow users to select known AGN, upload their own AGN catalogues, and select flaring events in both active and passive galaxies. This will support the science of tidal disruption events, changing look quasars, AGN flares, microlensing of background QSOs by foreground galaxies, and unusual long lived nuclear transients. Lasair will match radio and X-ray archival data with optical spectra, and the LSST lightcurves. Users will be able to select on these criteria or upload their own watch list to Lasair to combine with lightcurve parameters.

6.5 Milky Way and Local Group stellar transients

Within the TVS Science Collaboration most science for variables (typically recurrent and periodic signals) will be achieved with the annual data releases. However there is great opportunity in combining alerts with the data releases. Users can discover outbursts or large amplitude variability through the alerts and link to the data releases and full multi-year lightcurves. Lasair-ZTF currently can provide streams of objects matched to known stars (via watch lists of 10^6 objects) and trigger on a particular magnitude variability index. We are working with scientists within TVS in particular to define features that can be measured on the incoming stream and used to provide alerts. For example, outbursts of AM CVn stars\cite{AMCVn} which are then linked to the binary system's long term lightcurve (SDSS J1240-0159 is a recent example). Lasair-LSST will expand on its current functionality to provide seamless cross-links to the data releases within the UK IDAC infrastructure.

6.6 Solar System objects

LSST will provide an unprecedented opportunity for time-domain Solar System science.

6.7 New types of transient

In the future, we can expect surprises. The Lasair community has active groups dedicated to finding and following rare events such as superluminous supernovae, tidal disruption events, compact stellar mergers and black-hole forming supernovae.

In the future we expect further exotica to emerge, and it is the flexibility of Lasair's design which will allow relevant information to be streamed in, joint queries to be built and executed in real time, and follow-up facilities alerted and activated.

Please start here to discover the key ideas of Lasair:

CHAPTER

SEVEN

OBJECTS AND SOURCES

Lasair deals in *objects* and *sources*. A source is a detection by the telescope of an object.

A source is a collection of pixels on the telecsopes light-collection device, which is significantly brighter than it was in the reference imagery, that was taken at the beginning of the survey.

A source is detected with a specific narrowband optical filter:

LSST uses filters u,g,r,i,z and ZTF uses g,r.

When a lot of sources are found in the same place in the sky, the collection is called an object.

Thus an object is a star or similar that *does not move* in the sky.

Usually it is assumed that an object corresponds to a real astrophysical object, such as star or something extragalactic.

The brightness of a source in a transient survey is actually a *difference* brightness.

If an object is a variable star, then its optical flux was measured before the survey – a reference flux – and the source detections is the difference, positive or negative, from this.

When brightness is expressed as magnitudes, this measurement has two parts: absolute value, converted to magnitudes, and a flag to indicate positive or negative difference.

There are also solar-system objects and solar-system sources. The sources correspond to detections, and the objects to asteroids or other moving bodies in our solar system. However, the association of sources is mor difficult becuase that are in different positions in the sky due to orbital motion.

CHAPTER

EIGHT

LIGHTCURVE

A lightcurve is a record of the brightness of an astrophysical object with time, so it is a collection of (time, brightness) pairs. For LSST the brightness is measured as flux in nanoJanskies (nJ), and for ZTF it is measured with the traditional magnitude system. Note that the values in the lightcurve are *difference* fluxes, as defined in Objects and Sources.

Since each source brightness is measured for a specific optical filter, there may be several lightcurves for a given object, for example the g-lightcurve and r-lightcurve will be derived from the detections in the g filter and r filter respectively.

The nature of the lightcurve informs us of the underlying astrophysics. Variable stars can be characterised by the shape of their lightcurves, and explosive transients such as supernovae can be distinguished by the rise and fall rates of their lightcurves.

CHAPTER NINE

SKY CONTEXT

There are already a large number of astronomical catalogues, each containing carefully curated data about astronomical objects: stars, galaxies, variable stars, cataclysmic variables, active galactic nuclei, etc. When an existing object brightens, or a new object appears, astronomers want to know if it is already known, and if so, what kind of object it is. If the astronomer is searching for extra-galactic explosive events such as supernovae, they are usually associated with a galaxy. That astronomer is also interested if the explosive event has already been seen and registered by somebody else.

Lasair provides several kinds of sky context:

- Sherlock: A software package and integrated massive database system that provides a rapid and reliable spatial cross-match service for any astrophysical variable or transient. Details here.
- Transient Name Server (TNS) is the official IAU mechanism for reporting new astronomical transients such as supernova candidates. Once spectroscopically confirmed, new supernova discoveries are officially designated a SN name. Lasair keeps a cache of the database, updated every few hours. Details here.
- Personal Watchlists: Lasair allows users to upload personal catalogues of interesting sources, which are crossmatched in real time with incoming alerts.

CHAPTER

TEN

QUERIES AND FILTERS

Lasair attempts to blur the line between a *select query* and a *streaming filter*.

10.1 Select Query

The select query can be initiated through the Lasair web, or by using the Lasair API;

it has a `SELECT` clause and a `WHERE` clause that are entered separately, the first being

what is reported back and the second the criteria. There is also a choice of which data sources

to choose from in addition to the `object` table.

If I choose my own watchlist in addition to the `object` table, and the `SELECT` clause is

```
objects.objectId, watchlist_hits.name, objects.glatmean
```

and there the `WHERE` clause is

```
glatmean > 20
ORDER BY glatmean
```

and I click 'Run this Query', then I get a list of objects that are coincident with

the given watchlist, together with the watchlist's name, and the galactiv latitude.

The `WHERE` clause has restricted the results by galactic latitude, and the results

come in order of galactic latitude.

10.2 Multiple Tables

Lasair supports queries that join multiple tables, for example a watchlist of

your favourite sources, or the TNS list of known

supernovae and other transients. In this case, you are selecting **ONLY** those

objects that are **ALSO* in the chosen table. If you make a filter that selects `objectId` and you also choose a watchlist, then your filter returns only alerts

coincident with the sources in the watchlist.

10.3 Streaming Filter

The query-building page has a checkbox that changes a saved query to a filter, meaning that
whenever an incoming alert satisfies the criteria, a message is sent to the user of that
query. The message can be via email (with messages bundled into a 24-hour digest), or it
can be machine-readable by a Kafka stream. In this way, a user – or their machine – can be
alerted in near-real-time, withing minutes of the telescope taking the data. This message is
repeated whenever new data comes in; in the example above, a message wouyld be generated every
time an alert coincides with the watchlist and has `glatmean > 20`.
However, the results of a streaming filter are not identical to running the same stored
query from web or API. As noted above, a given object can be reported multiple times when
the streaming filter is operating, but only once in the select query. The other difference
is that the ordering of results from a streaming filter will *always* by time order, so the `ORDER BY` part of the `WHERE` clause is ignored.

10.4 Cookbook

For instructions on how to make a filter, see Make a Lasair Filter.

CHAPTER ELEVEN

CODING WITH LASAIR

There is a python client for Lasair .

Its usage is explained in the page Lasair API and Client.

There are a number of python notebooks that show how

to use the Lasair client.

CHAPTER

TWELVE

LASAIR'S ADDED VALUE

1. Sherlock
2. Lightcurve features
3. External brokers

CHAPTER

THIRTEEN

ANNOTATIONS

Lasair allows users to add information to the database, that can then be used
as part of a query by another user. Each *annotation* is associated with a
specific Lasair object, and with a specific *annotator*, and may
contain:

- `objectId`: the Lasair object being annotated
- `topic`: the name of the annotator that produced this annotation
- `classification`: a short string drawn from a fixed vocabulary, eg "kilonova".
- `explanation`: a natural language explanation of the classification, eg "probable kilonova but could also be supernova"
- `classjson`: the annotation information expressed as a JSON dictionary
- `url`: a URL where more information can be obtained, for example
  a spectrum of the object obtained by follow-up.

The `classification` is easy to query: it is just a word; but the `classjson`
can hold complex information and querying is more sophisticated.
Annotations can be pushed to the Lasair database using the Lasair client,
however the user must be authenticated to do so. Lasair staff are happy to
receive a request to create an annotator, and the successful user
will be given a `topic` name that allows them to upload annotations.

13.1 Cookbook

For instructions on how to run your own annotator, see Making an Annotator.
Lasair's core features are described in more detail here:

CHAPTER

FOURTEEN

SKY SEARCH

At the top of every Lasair page is a form that can be filled in to do various kinds of search. The following are supported:

- Search by RA and Dec decimal degrees, delimited by space or comma:

```
308.590715 9.278195
308.590715, 9.278195
```

- Search by RA and Dec sexagesimal coordinates, delimited by space or comma:

```
20:34:21.7  09:16:41.5
20:34:21.7, 09:16:41.5
```

- Adding a search radius in arcseconds to any of the previous, for example:

```
308.590715 9.278195 60
```

- Search by object identifier, for example `ZTF23aacvrxx`
- Search by TNS identifier, for example `AT2020iry` or `SN2020iry` or `2020iry`.

CHAPTER

FIFTEEN

SHERLOCK (SKY CONTEXT)

Detections in the input data stream that have been aggregated into *objects*

(i.e. groups of detections) and identified as static transients (i.e. not moving objects)

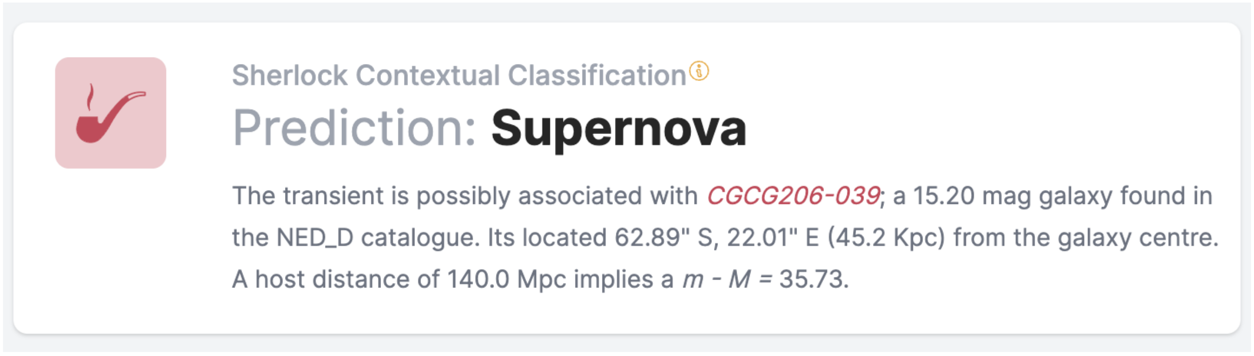
are spatially context classified against a large number of archival sources

(e.g. nearby galaxies, known CVs, AGNs, etc).

The information derived from this context check is injected as an object annotation

The software used is called *Sherlock*

and is discussed below. Here is how the Sherlock information looks on the Lasair object page:



Sherlock Contextual Classification

Prediction: **Supernova**

The transient is possibly associated with *CGCG206-039*; a 15.20 mag galaxy found in the NED_D catalogue. Its located 62.89" S, 22.01" E (45.2 Kpc) from the galaxy centre. A host distance of 140.0 Mpc implies a *m* - *M* = 35.73.

This panel shows a natural language diescription of the association, derived from

some of the fields of the Sherlock table:

-|`classification`: can be any of the 9 strings: NULL, AGN, BS, CV, NT, ORPHAN, SN, UNCLEAR, VS;

in the image above the acronym 'SN' has been expanded to 'Supernova', which means

that the Lasair object is strongly associated with a host galaxy, but not so close as 'NT', which is

Nuclear Transient.

-|`catalogue_object_id`: is the name of associated galaxy, in this case CGCG206-039.

Not shown in the panel is the `catalogue_table_name`, which is the catalogue (namespace) of

that name. In this case the name is in the NED data system.

-|`northSeparationArcsec` and `eastSeparationArcsec` for the angualr separation between the

transient and the centre of the associated galaxy. This is translated to `physical_separation_kpc`

(separation in kilo-parsecs), using `direct_distance` which is mega-parsecs – if available.

-|`Mag` is the magnitude of the associated galaxy, using the system in `MagFilter`. This can be combined with distance information to derive absolute magnitude.

The full schema for the Sherlock table is in the Lasair Schema Browser.

15.1 How does Sherlock work?

Sherlock is a software package and integrated massive database system that

provides a rapid and reliable spatial cross-match service for any astrophysical

variable or transient. The concept originated in the PhD thesis of D. Young at QUB, and has

been developed by Young et al. in many iterations and cycles since. It associates the

position of a transient with all major astronomical catalogues and assigns a basic

classification to the transient. At its most basic, it separates stars, AGN and

supernova-like transients. It has been tested within QUB on a daily basis with

ATLAS and Pan-STARRS transients, and within PESSTO as part of the PESSTO

marshall system that allows prioritising of targets. It is thus a boosted

decision tree algorithm. A full paper describing the code, catalogues and

algorithms is in preparation (Young et al. in prep). A summary is included in Section 4.2 of

"Design and Operation of the ATLAS Transient Science Server"

(Smith, Smartt, Young et al. 2020, submitted to PASP: https://arxiv.org/abs/2003.09052).

We label the current version as the official release of Sherlock 2.0.

The major upgrade from previous versions are that it includes Pan-STARRS DR1

(including the Tachibana & Miller 2018 star-galaxy separation index) and

Gaia DR2 catalogues, along with some adjustments to the ranking algorithm.

A boosted decision tree algorithm (internally known as *Sherlock*) mines a library of historical and on-going astronomical survey data and attempts to predict the nature of the object based on the resulting crossmatched associations found. One of the main purposes of this is to identify variable stars, since they make up about 50% of the objects, and to associate candidate extragalactic sources with potential host galaxies. The full details of this general purpose algorithm and its implementation will be presented in an upcoming paper (Young et al. in prep), and we give an outline of the algorithm here.

The library of catalogues contains datasets from many all-sky surveys such as

- Gaia DR1 and DR2 (Gaia Collaboration et al. 2016, 2018),
- Pan-STARRS1 Science Consortium surveys (Chambers et al. 2016; Magnier, Chambers, et al. 2016; Magnier, Sweeney, et al. 2016; Magnier, Schlafly, et al. 2016; Flewelling et al. 2016) and the catalogue of probabilistic classifications of unresolved point sources by (Tachibana and Miller 2018) which is based on the Pan-STARRS1 survey data.
- The SDSS DR12 PhotoObjAll Table, SDSS DR12 SpecObjAll Table (Alam et al. 2015) contains both reliable star-galaxy separation and photometric redshifts which are useful in transient source classification.

Extensive catalogues with lesser spatial resolution or colour information that we use are

- GSC v2.3 (Lasker et al. 2008) and

- 2MASS catalogues (Skrutskie et al. 2006).

Sherlock employs many smaller source-specific catalogues such as

- Million Quasars Catalog v5.2 (Flesch 2019),
- Veron-Cett AGN Catalogue v13 (Véron-Cetty and Véron 2010),
- Downes Catalog of CVs (Downes et al. 2001),
- Ritter Cataclysmic Binaries Catalog v7.21 (Ritter and Kolb 2003).

For spectroscopic redshifts we use the

- GLADE Galaxy Catalogue v2.3 (Dálya et al. 2018) and the
- NED-D Galaxy Catalogue v13.1

Sherlock also has the ability to remotely query the NASA/IPAC Extragalactic Database, caching results locally to speed up future searches targeting the same region of sky, and in this way we have built up an almost complete local copy of the NED catalogue. More catalogues are continually being added to the library as they are published and become publicly available.

At a base-level of matching *Sherlock* distinguishes between transient objects *synonymous* with (the same as, or very closely linked, to) and those it deems as merely *associated* with the catalogued source. The resulting classifications are tagged as *synonyms* and *associations*, with synonyms providing intrinsically more secure transient nature predictions than associations. For example, an object arising from a variable star flux variation would be labeled as *synonymous* with its host star since it would be astrometrically coincident (assuming no proper motion) with the catalogued source. Whereas an extragalactic supernova would typically be *associated* with its host galaxy - offset from the core, but close enough to be physically associated. Depending on the underpinning characteristics of the source, there are 7 types of predicted-nature classifications that Sherlock will assign to a transient:

1. **Variable Star** (VS) if the transient lies within the synonym radius of a catalogued point-source,
2. **Cataclysmic Variable** (CV) if the transient lies within the synonym radius of a catalogued CV,
3. **Bright Star** (BS) if the transient is not matched against the synonym radius of a star but is associated within the magnitude-dependent association radius,
4. **Active Galactic Nucleus** (AGN) if the transient falls within the synonym radius of catalogued AGN or QSO.
5. **Nuclear Transient** (NT) if the transient falls within the synonym radius of the core of a resolved galaxy,
6. **Supernova** (SN) if the transient is not classified as an NT but is found within the magnitude-, morphology- or distance-dependant association radius of a galaxy, or
7. **Orphan** if the transient fails to be matched against any catalogued source.

For Lasair the synonym radius is set at 1.5. This is the crossmatch-radius used to assign predictions of VS, CV, AGN and NT. The process of attempting to associate a transient with a catalogued galaxy is relatively nuanced compared with other crossmatches as there are often a variety of data assigned to the galaxy that help to greater inform the decision to associate the transient with the galaxy or not. The location of the core of each galaxy is recorded so we will always be able to calculate the angular separation between the transient and the galaxy. However we may also have measurements of the galaxy morphology including the angular size of its semi-major axis. For Lasair we reject associations if a transient is separated more than 2.4 times the semi-major axis from the galaxy, if the semi-major axis measurement is available for a galaxy. We may also have a distance measurement or redshift for the galaxy enabling us to convert angular separations between transients and galaxies to (projected) physical-distance separations. If a transient is found more than 50 Kpc from a galaxy core the association is rejected.

Once each transient has a set of independently crossmatched synonyms and associations, we need to self-crossmatch these and select the most likely classification. The details of this will be presented in a future paper (Young et al. in prep). Finally the last step is to calculate some value added parameters for the transients, such as absolute peak magnitude if a distance can be assigned from a matched catalogued source, and the predicted nature of each transient is presented to the user along with the lightcurve and other information.

We have constructed a multi-billion row database which contains all these catalogues. It currently consumes about 4.5TB and sits on a separate, similarly specified machine to that of the Lasair database. It will grow significantly as new catalogues are added (e.g. Pan-STARRS 3__ DR2, VST and VISTA surveys, future Gaia releases etc).

The *Sherlock* code is open source and can be found at: https://github.com/thespacedoctor/sherlock. Documentation is also available online here: https://qub-sherlock.readthedocs.io/en/stable/.

Although the code for *Sherlock* is public, it requires access to a number of large databases which are custom built from their original, public, releases. The latter is proprietary and therefore would require some effort from users to reproduce. As part of the Lasair project we are exploring public access to the integrated *Sherlock* code and database information through an API.

Sherlock 2.0 was reviewed as a LSST:UK Deliverable in March 2020. The review noted that an algorithm enhancement would be desirable to take into account stellar proper motions, since some proper motion stars will be variable and if cross-matched with a static catalogue will fall outside the nominal match radius. This is an enhancement we will taken forward for future versions.

15.2 Sherlock References

Alam, Shadab, Franco D Albareti, Carlos Allende Prieto, F Anders, Scott F Anderson, Timothy Anderton, Brett H Andrews, et al. 2015. “The Eleventh and Twelfth Data Releases of the Sloan Digital Sky Survey: Final Data from SDSS-III.” *The Astrophysical Journal Supplement Series* 219 (1). IOP Publishing: 12. https://doi.org/10.1088/0067-0049/219/1/12.

Chambers, K. C., E. A. Magnier, N. Metcalfe, H. A. Flewelling, M. E. Huber, C. Z. Waters, L. Denneau, et al. 2016. “The Pan-STARRS1 Surveys.” *ArXiv E-Prints*, December.

Dálya, G, G Galgóczi, L Dobos, Z Frei, I S Heng, R Macas, C Messenger, P Raffai, and R S de Souza. 2018. “GLADE: A galaxy catalogue for multimessenger searches in the advanced gravitational-wave detector era.” *Monthly Notices of the Royal Astronomical Society* 479 (2): 2374–81. https://doi.org/10.1093/mnras/sty1703.

Downes, Ronald A, Ronald F Webbink, Michael M Shara, Hans Ritter, Ulrich Kolb, and Hilmar W Duerbeck. 2001. “A Catalog and Atlas of Cataclysmic Variables: The Living Edition.” *The Publications of the Astronomical Society of the Pacific* 113 (7): 764–68. https://doi.org/10.1086/320802.

Flesch, Eric W. 2019. “The Million Quasars (Milliquas) Catalogue, v6.4.” *arXiv.org*, December, arXiv:1912.05614. http://arxiv.org/abs/1912.05614v1.

Flewelling, H. A., E. A. Magnier, K. C. Chambers, J. N. Heasley, C. Holmberg, M. E. Huber, W. Sweeney, et al. 2016. “The Pan-STARRS1 Database and Data Products.” *ArXiv E-Prints*, December.

Gaia Collaboration, A. G. A. Brown, A. Vallenari, T. Prusti, J. H. J. de Bruijne, C. Babusiaux, C. A. L. Bailer-Jones, et al. 2018. “Gaia Data Release 2. Summary of the contents and survey properties” 616 (August): A1. https://doi.org/10.1051/0004-6361/201833051.

Gaia Collaboration, A. G. A. Brown, A. Vallenari, T. Prusti, J. H. J. de Bruijne, F. Mignard, R. Drimmel, et al. 2016. “Gaia Data Release 1. Summary of the astrometric, photometric, and survey properties” 595 (November): A2. https://doi.org/10.1051/0004-6361/201629512.

Lasker, Barry M, Mario G Lattanzi, Brian J McLean, Beatrice Bucciarelli, Ronald Drimmel, Jorge Garcia, Gretchen Greene, et al. 2008. “The Second-Generation Guide Star Catalog: Description and Properties.” *The Astronomical Journal* 136 (2). IOP Publishing: 735–66. https://doi.org/10.1088/0004-6256/136/2/735.

Magnier, E. A., K. C. Chambers, H. A. Flewelling, J. C. Hoblitt, M. E. Huber, P. A. Price, W. E. Sweeney, et al. 2016. “Pan-STARRS Data Processing System.” *ArXiv E-Prints*, December.

Magnier, E. A., E. F. Schlafly, D. P. Finkbeiner, J. L. Tonry, B. Goldman, S. Röser, E. Schilbach, et al. 2016. “Pan-STARRS Photometric and Astrometric Calibration.” *ArXiv E-Prints*, December.

Magnier, E. A., W. E. Sweeney, K. C. Chambers, H. A. Flewelling, M. E. Huber, P. A. Price, C. Z. Waters, et al. 2016. "Pan-STARRS Pixel Analysis : Source Detection & Characterization." *ArXiv E-Prints*, December.

Ritter, H, and U Kolb. 2003. "Catalogue of cataclysmic binaries, low-mass X-ray binaries and related objects (Seventh edition)." *Astronomy and Astrophysics* 404 (1). EDP Sciences: 301–3. https://doi.org/10.1051/0004-6361:20030330.

Skrutskie, M. F., R. M. Cutri, R. Stiening, M. D. Weinberg, S. Schneider, J. M. Carpenter, C. Beichman, et al. 2006. "The Two Micron All Sky Survey (2MASS)" 131 (February): 1163–83. https://doi.org/10.1086/498708.

Tachibana, Yutaro, and A. A. Miller. 2018. "A Morphological Classification Model to Identify Unresolved PanSTARRS1 Sources: Application in the ZTF Real-time Pipeline" 130 (994): 128001. https://doi.org/10.1088/1538-3873/aae3d9.

Véron-Cetty, M P, and P Véron. 2010. "A catalogue of quasars and active nuclei: 13th edition." *Astronomy and Astrophysics* 518 (July). EDP Sciences: A10. https://doi.org/10.1051/0004-6361/201014188.

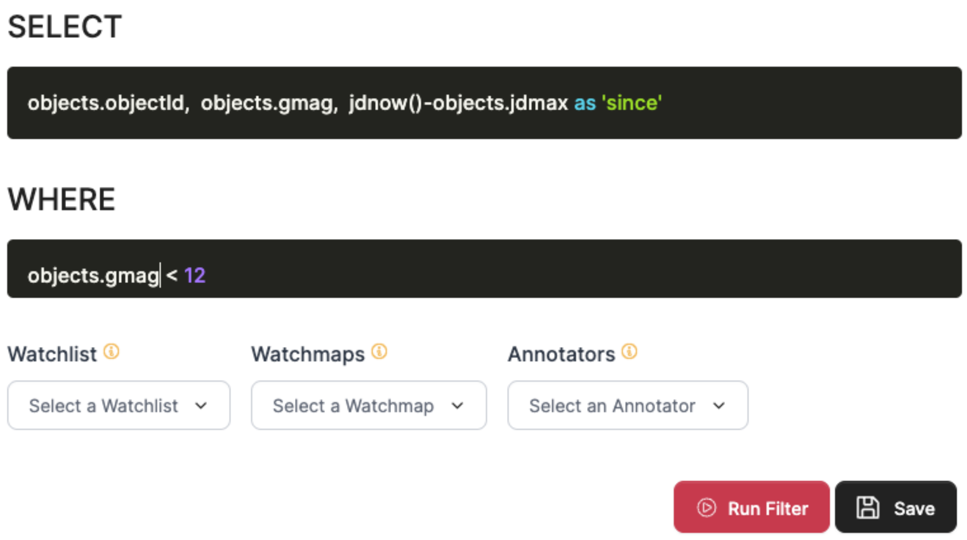
1. https://ned.ipac.caltech.edu/Library/Distances/

CHAPTER

SIXTEEN

MAKE A LASAIR FILTER

Lasair is built around "filters" of the alert stream. Users create a filter with SQL clauses, based on the attributes of the object associated with the alert: its lightcurve, sky context, etc.
First you make a filter and run it on the previous alerts, and then you can save the filter (if you have a Lasair account). You can convert your filter to a *streaming* filter, so that results are sent to your email or to your own machine as soon as they are available.
Let us make a filter for alerts associated with bright stars. Click on "Filters" in the left margin, then "Create New" at top right.


SELECT

```
objects.objectId,  objects.gmag,  jdnow()-objects.jdmax as 'since'
```

WHERE

```
objects.gmag < 12
```

Fill in the form as shown here. Name and description, then check the "email" box, and fill in the SELECT as

```
objects.objectId, objects.gmag, jdnow()-objects.jdmax AS since
```

and fill in the WHERE as

```
objects.gmag < 12
```

If you click "Run Filter", all the objects will be returned that are brighter than 12th magnitude and the brightness is variable.
Perhaps you would like to see the obejcts with the most recent alerts first:
just add to the WHERE clause the phrase

```
ORDER BY objects.jdmax DESC
```

and click "Run Filter" again. The attribute `jdmax` of an object is the Julian Day

of the most recent detection of that object.

A good way to understand how filters are made is to browse those in the Public Gallery.

Looking at the Lasair Schema Browser, there are several additional tables that you can use in addition to the `objects` table:

- `sherlock_classifications`: the sky context.
- `crossmatch_tns`: the crossmatch with the Transient Name Seerver.
- A watchlist of your choice, either your own or one from the public gallery
- A watchmap of your choice, either your own or one from the public gallery
- An annotator of your choice, either your own or one from the public gallery

16.1 Copying a Filter

An easy way to get started with building a filter is to copy an existing one; either your own or

from the public gallery. You will be asked to provide a name for the new filter, which

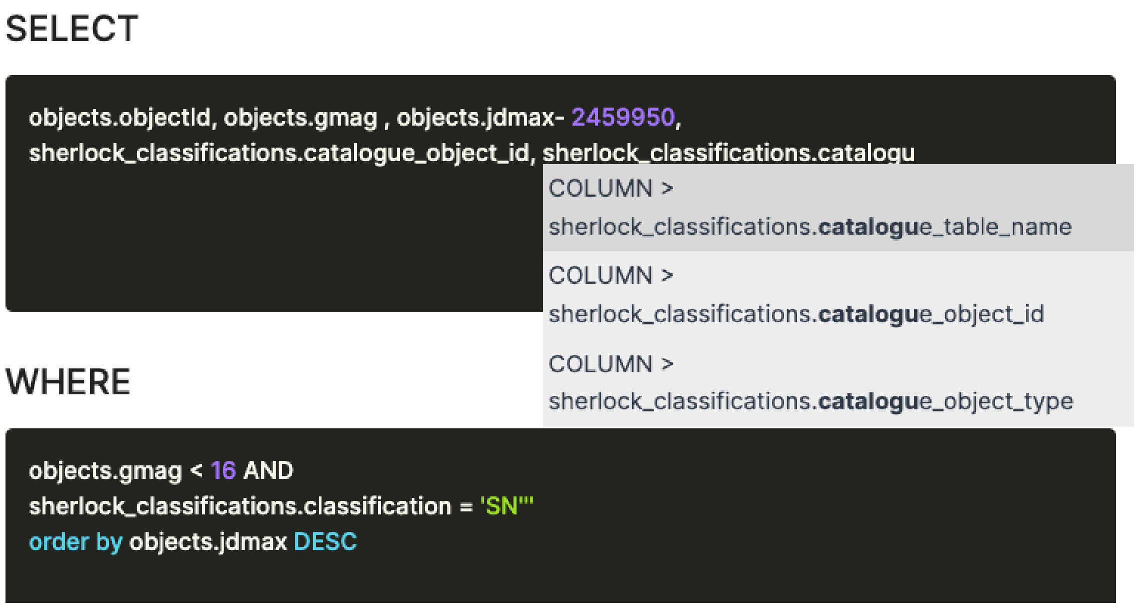
should be different from the original. You can then modify the SQL code and save it.

16.2 Filtering on Sky Context

Once you start typing into the SELECT or WHERE boxes, the autocomplete function

is activated. For example if you want to know the host galaxy associated

with an object, you type 'sherlock' and the options show – the grey box below.



SELECT

```
objects.objectId, objects.gmag , objects.jdmax- 2459950,
sherlock_classifications.catalogue_object_id, sherlock_classifications.catalogu
```

COLUMN >
sherlock_classifications.**catalogu**e_table_name

COLUMN >
sherlock_classifications.**catalogu**e_object_id

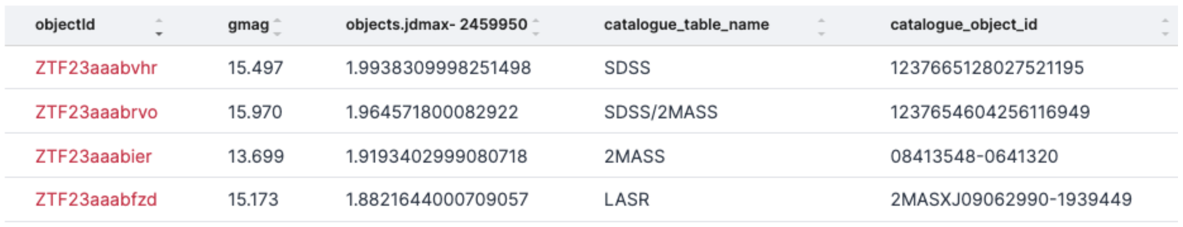
COLUMN >
sherlock_classifications.**catalogu**e_object_type

WHERE

```
objects.gmag < 16 AND
sherlock_classifications.classification = 'SN'"
order by objects.jdmax DESC
```

The name of the galaxy is the id and the catalogue table name to which

that id applies. Here are some sample results fromn the query above:



objectId	gmag	objects.jdmax- 2459950	catalogue_table_name	catalogue_object_id
ZTF23aaabvhr	15.497	1.9938309998251498	SDSS	1237665128027521195
ZTF23aaabrvo	15.970	1.964571800082922	SDSS/2MASS	1237654604256116949
ZTF23aaabier	13.699	1.9193402999080718	2MASS	08413548-0641320
ZTF23aaabfzd	15.173	1.8821644000709057	LASR	2MASXJ09062990-1939449

The TNS table `crossmatch_tns` operates in a similar way to the `sherlock_classificaitons` table. See the Schema Browser for details of the available attributes.

16.3 Filtering on a Watchlist/Watchmap

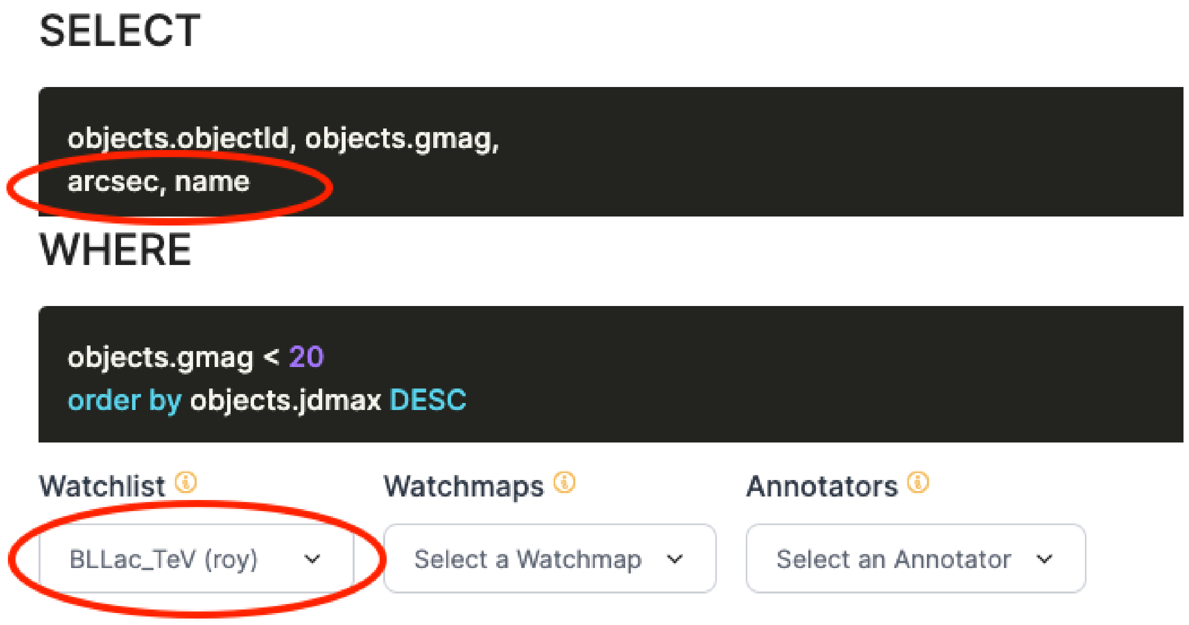
You can select on objects coincident with a watchlist, either your own

or one from the public gallery. At the bottom of the filter creation form

is a selection of watchlists (red oval below). You can then also

choose the attributes `name` and `arcsec`, the name and angular distance of the

coincident source from the watchlist.



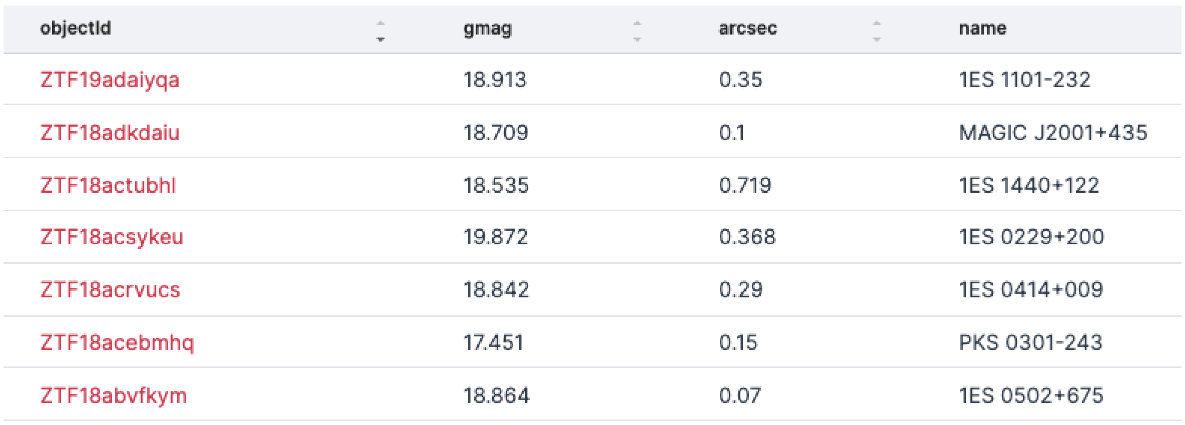
SELECT

```
objects.objectId, objects.gmag,
arcsec, name
```

WHERE

```
objects.gmag < 20
order by objects.jdmax DESC
```

The results might look something like this:



objectId	gmag	arcsec	name
ZTF19adaiyqa	18.913	0.35	1ES 1101-232
ZTF18adkdaiu	18.709	0.1	MAGIC J2001+435
ZTF18actubhl	18.535	0.719	1ES 1440+122
ZTF18acsykeu	19.872	0.368	1ES 0229+200
ZTF18acrvucs	18.842	0.29	1ES 0414+009
ZTF18acebmhq	17.451	0.15	PKS 0301-243
ZTF18abvfkym	18.864	0.07	1ES 0502+675

The watchmaps operate in a similar way with the selection box, but there are no

attributes available.

16.4 Filtering on an Annotator

When an annotation is uploaded to Lasair, it belongs to a specific object,

and it has attributes:

- `topic`: the name of the annotator, generally associated with a specific user who is responsible.
- `classification`: a short string drawn from a fixed vocabulary, eg "kilonova".
- `explanation`: a natural language explanation of the classification, eg "probable kilonova but could also be supernova"
- `classjson`: the annotation expressed as a JSON dictionary
- `url`: a URL where more information can be obtained about the classification of this object

These can be selected and used in the query as with the watchlist/watchmap,

except for the `classjson` attribute which is different and more complex.

While most attributes in SQL are strings or numbers, the `classjson` can

contain lists and dictionaries. If for example the `classjson` is a dictionary

with keys `color` and `size`, then this clause will extract the color

when the size is greater than 4:

```
SELECT JSON_EXTRACT(jdoc, '$.color') AS color FROM jtable
WHERE
JSON_EXTRACT(jdoc, '$.size') > 4;
```

CHAPTER

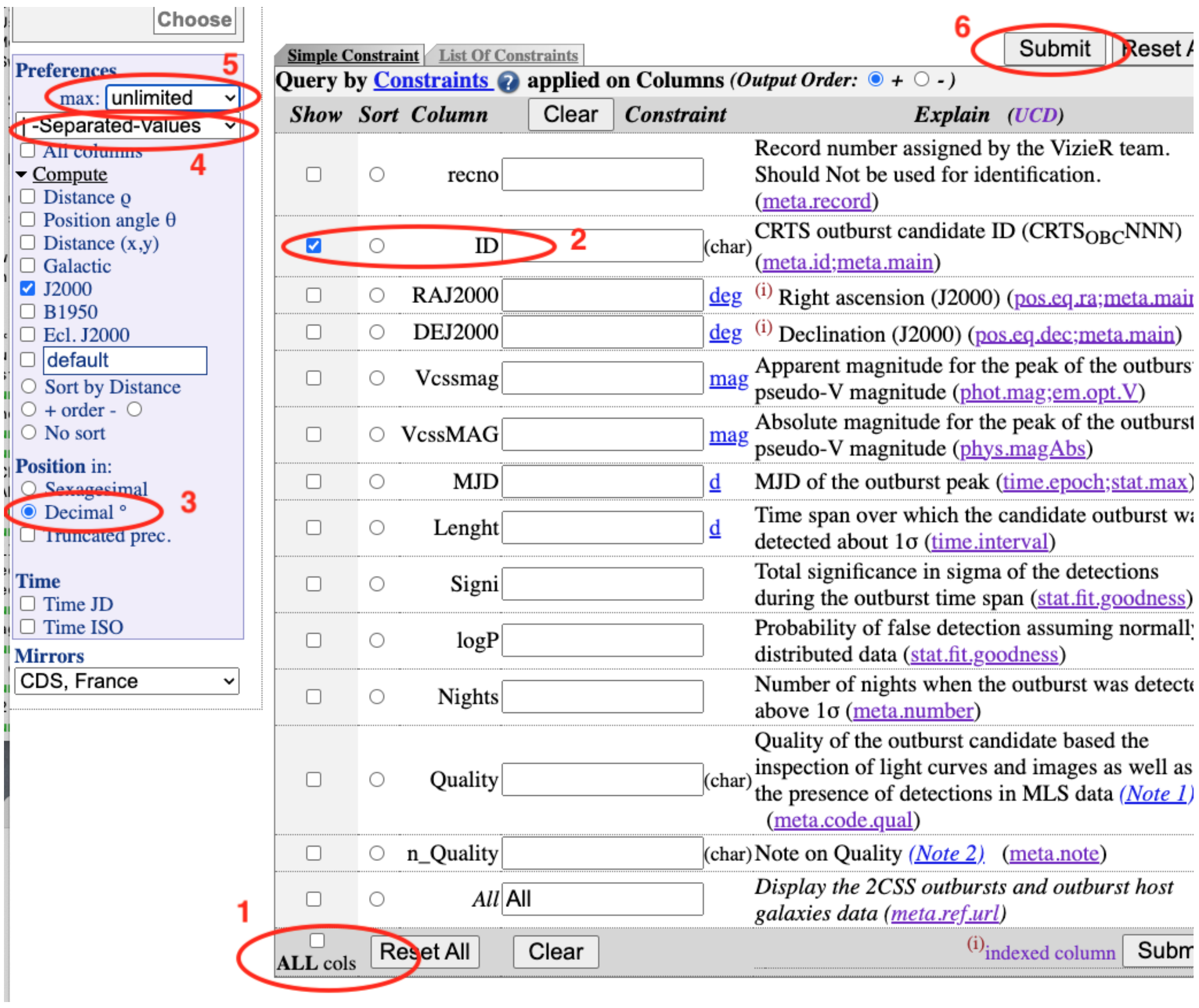
SEVENTEEN

WATCHLISTS

A watchlist is a set of points in the sky, together with a radius in arcseconds, which
can be the same for all sources, or different for each.
It is assumed to be a list of "interesting" sources, so that any transient that
falls within the radius of one of the sources might indicate activity of that source.
Each user of the Lasair system has their own set of watchlists, and can be
alerted when a transient is coincident with a watchlist source. Here, the word coincident means
within the radius of the source.
An "Active" watchlist is one that is run every day, so that it is up to date with the latest objects.

17.1 Create new watchlist

You can create a watchlist of sources by preparing a text file, where each
comma-separated or |-separated line has RA and Dec in decimal degrees,
an identifier, with optional radius in arcseconds. One way to do this is
with Vizier (see below) or with a spreadsheet
program such as Excel or Numbers.
Here is an example of the data. The 42 entries are *BL Lac candidates for TeV observations (Massaro+, 2013)*
Note that you must be logged in to create a watchlist.
Many astronomers are interested in transients that are associated with specific
astronomical objects, perhaps active galaxies or star formation regions.
Once you have an account on Lasair, you can create any number of watchlists, to be
used in the query engine. To be specific, suppose we are interested in the 42 objects in the
catalogue BL Lac candidates for TeV observations (Massaro+, 2013),
that can be found in the Vizier library of catalogues. You can make your
watchlist "public", so other Lasair users can see it and use it in queries,
and you can make your watchlist "active", meaning that the crossmatch (see below)
is done automatically every day.
The following is how to make the correct file format from Vizier.



Preferences

max: unlimited

|-Separated-Values

☐ All columns

Compute
- ☐ Distance ϱ
- ☐ Position angle θ
- ☐ Distance (x,y)
- ☐ Galactic
- ☑ J2000
- ☐ B1950
- ☐ Ecl. J2000
- ☐ default
- ○ Sort by Distance
- ○ + order - ○
- ○ No sort

Position in:
- ○ Sexagesimal
- ● Decimal °
- ☐ Truncated prec.

Time
- ☐ Time JD
- ☐ Time ISO

Mirrors

CDS, France

Query by Constraints applied on Columns *(Output Order:* ● + ○ - *)*

Show	*Sort*	*Column*	*Constraint*		*Explain (UCD)*
☐	○	recno			Record number assigned by the VizieR team. Should Not be used for identification. (meta.record)
☑	○	ID		(char)	CRTS outburst candidate ID (CRTS$_{OBC}$NNN) (meta.id;meta.main)
☐	○	RAJ2000		deg	(i) Right ascension (J2000) (pos.eq.ra;meta.main
☐	○	DEJ2000		deg	(i) Declination (J2000) (pos.eq.dec;meta.main)
☐	○	Vcssmag		mag	Apparent magnitude for the peak of the outburs pseudo-V magnitude (phot.mag;em.opt.V)
☐	○	VcssMAG		mag	Absolute magnitude for the peak of the outburst pseudo-V magnitude (phys.magAbs)
☐	○	MJD		d	MJD of the outburst peak (time.epoch;stat.max)
☐	○	Lenght		d	Time span over which the candidate outburst wa detected about 1σ (time.interval)
☐	○	Signi			Total significance in sigma of the detections during the outburst time span (stat.fit.goodness)
☐	○	logP			Probability of false detection assuming normall distributed data (stat.fit.goodness)
☐	○	Nights			Number of nights when the outburst was detect above 1σ (meta.number)
☐	○	Quality		(char)	Quality of the outburst candidate based the inspection of light curves and images as well as the presence of detections in MLS data (Note 1) (meta.code.qual)
☐	○	n_Quality		(char)	Note on Quality (Note 2) (meta.note)
☐	○	*All*	All		*Display the 2CSS outbursts and outburst host galaxies data (meta.ref.url)*

☐ ALL cols

(i)indexed column

First you select a catalogue, which may consist of a number of tables. Select JUST ONE TABLE, so that there is just a single list of attributes. For example, this link has two tables, but this link is for a single table.

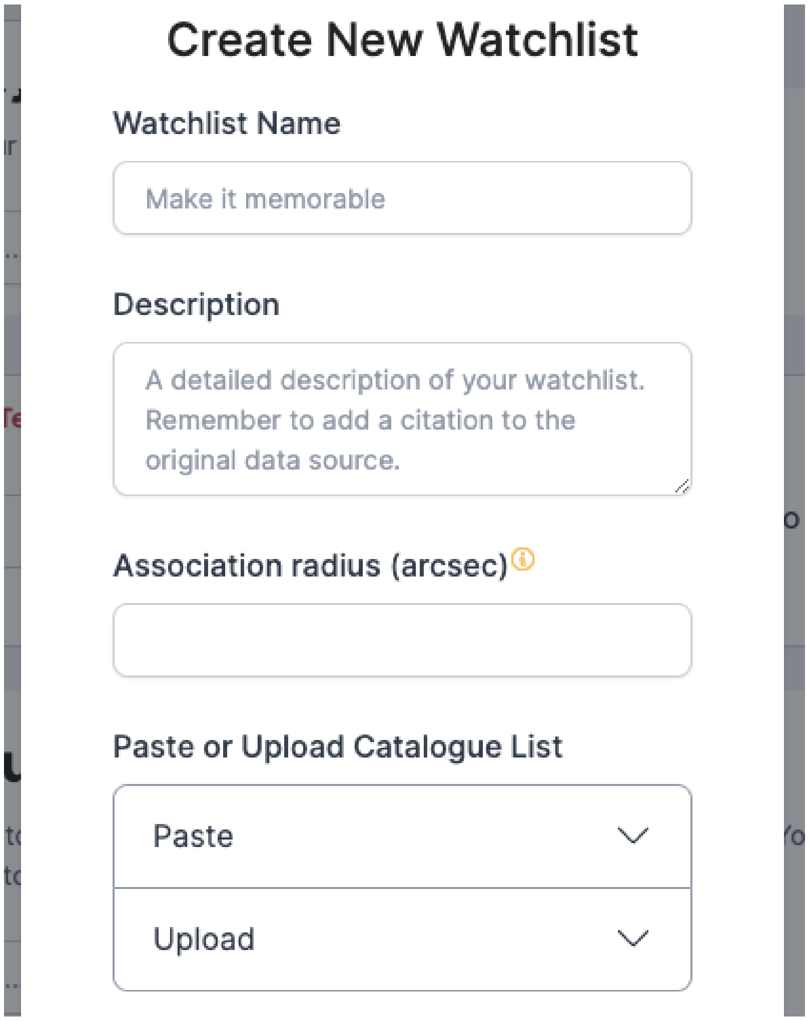
Once you have selected your table,

1. Deselect all the columns
2. Select a column that can act as the identifier for each source.

   These need to be unique and not empty: if not, you must edit the resulting file to make it so.
3. Choose "Decimal" for the coordinates
4. Choose "|-separated" for the format
5. Select "unlimited" or however many you want in your watchlist
6. Click submit to download the file.

Once you have the file, you can paste it into a form, or upload the file directly.

There may be error messages about unparsable lines, which can be eliminated by editing the file so every non-numerical line begins with the # symbol.



Create New Watchlist

Watchlist Name

Make it memorable

Description

A detailed description of your watchlist. Remember to add a citation to the original data source.

Association radius (arcsec)

Paste or Upload Catalogue List

Paste

Upload

The upload form is shown here:

Fill in the name and description of the watchlist. Choose a default value of the radius to use in matching, in arcseconds.

Each line should be RA, Dec, ID, and may have a fourth entry, the radius to use in matching, in arcseconds, if different from the default. Then click "Create".

Here is a successful creation of a watchlist. Some messages – "Bad line" – because there were some lines without data, but you can ignore these, and look for where it says "Watchlist created successfully". You can now find it in the list of "My Watchlists".

17.2 Find outbursts from my watchlist

Once you have made a watchlist, you may be interested in being notified whenever
something unusual – outburst for example – happens to one of your sources.
Thus we combine a watchlist with a query on magnitude that detects fast rise.
For the watch list see Build a Watchlist of your sources, and for the query we
utilise the moving averages of apparent magnitudes that Lasair provides.

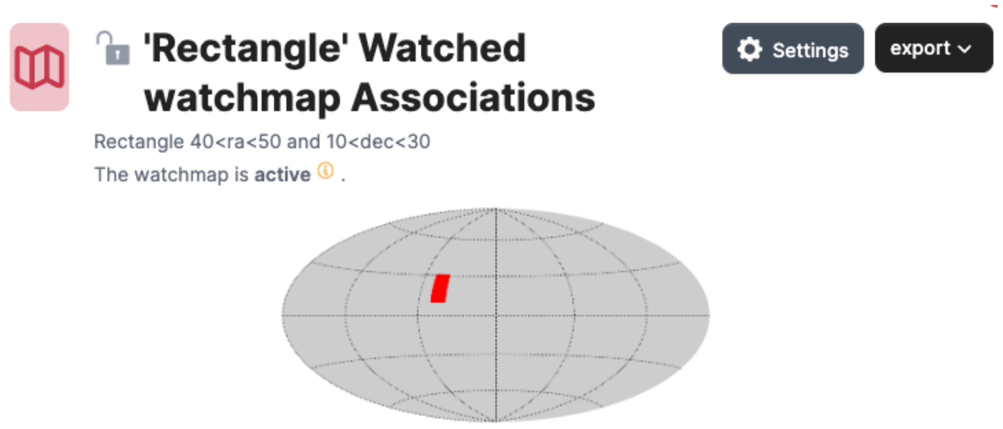
CHAPTER

EIGHTEEN

WATCHMAPS (SKY REGIONS)

18.1 What is it?

A watchmap is a specification of an area of the sky, that can be used as part of a Lasair filter.
An example might be the footprint of another survey, or the area of sky where a multimessenger
event occured. If the watchmap has been set to "active", then all alerts ingested to Lasair are tested
against it, and tagged if inside the watchmap. A filter can then be built that selects only
alerts falling inside the watchmap.

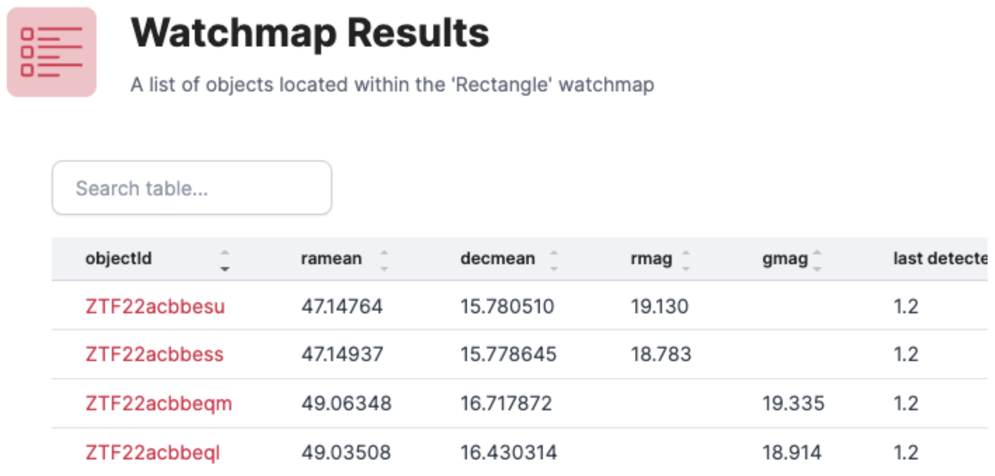


'Rectangle' Watched watchmap Associations

Rectangle 40<ra<50 and 10<dec<30

The watchmap is **active** .

Above we see a simple watchmap displayed as a Mollweide projection
on the sky. It is a rectangle of sky with vertices (40,10), (50,10), (50, 30), (40,30).
The watchmap can also have a name and description, accessible from the "settings" button.
Also here, the owner can choose for the watchmap to be public or not,
and for the watchmap to be "active" or not.



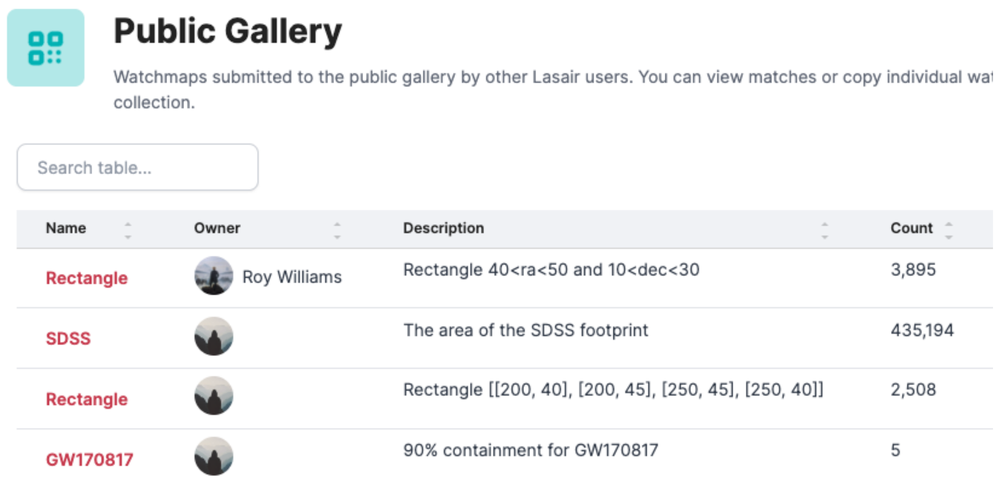
Watchmap Results

A list of objects located within the 'Rectangle' watchmap

Search table...

objectId	ramean	decmean	rmag	gmag	last detecte
ZTF22acbbesu	47.14764	15.780510	19.130		1.2
ZTF22acbbess	47.14937	15.778645	18.783		1.2
ZTF22acbbeqm	49.06348	16.717872		19.335	1.2
ZTF22acbbeql	49.03508	16.430314		18.914	1.2

Below the Mollweide picture of the watchmap is a partial list of the Lasair alerts that fall inside it, together with magnitudes. The list is sorted so those observed most recently are first.



Public Gallery

Watchmaps submitted to the public gallery by other Lasair users. You can view matches or copy individual wat collection.

Search table...

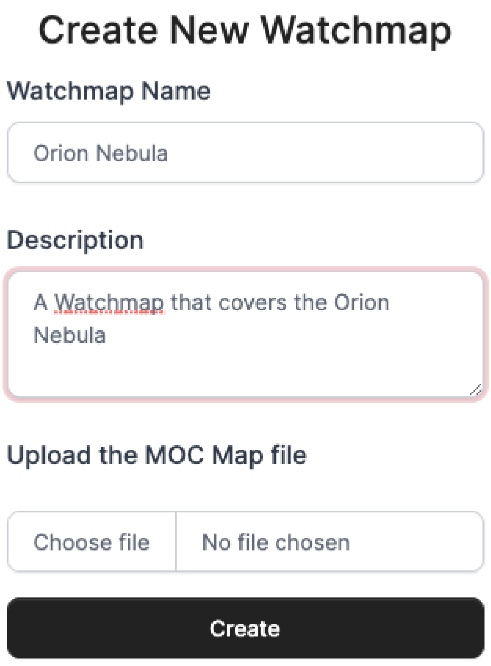
Name	Owner	Description	Count
Rectangle	Roy Williams	Rectangle 40<ra<50 and 10<dec<30	3,895
SDSS		The area of the SDSS footprint	435,194
Rectangle		Rectangle [[200, 40], [200, 45], [250, 45], [250, 40]]	2,508
GW170817		90% containment for GW170817	5

Here we see how the gallery of public watchmaps may look. Each show name, description, and how many alerts fall within it.

18.2 How can I make one?

Building a watchmap starts with building a MOC file – see the information at

https://cds-astro.github.io/mocpy/.



This is the dialogue to load a MOC file, in this case to cover the Orion Nebula.

It expects to upload a MOC file.

One way to build this from python is this code:

Create a MOC from a Concave Polygon

For the Orion Nebula, we choose a rectangle like this:

```
import astropy.units as u
import numpy as np
from astropy.coordinates import Angle, SkyCoord
from mocpy import MOC, WCS

vertices = np.array([
    [83.4, -5.0],
    [84.1, -5.0],
    [84.1, -5.7],
    [83.4, -5.7]])
skycoord = SkyCoord(vertices, unit="deg", frame="icrs")
moc = MOC.from_polygon_skycoord(skycoord, max_depth=8)
moc.write("polygon_moc.fits", format="fits", overwrite=True)
```

and we may adjust the `max_depth`: smaller values give smaller files, but larger values

give more accurate edges. For a fluffy thing like the Orion Nebula, `max_depth=8` is quite

sufficient. The output of this program is a file called `polygon_moc.fits`, that can be uploaded.

Note that the new watchmap is by default public and active.

It is only future alerts that will be matched against the new watchmap; however if you contact us, quoting the URL for the watchmap, we can

match it against past alerts.

CHAPTER

NINETEEN

LASAIR API AND CLIENT

The Lasair-Sherlock API allows developers to run queries and cone-searches, to see outputs from streaming queries, and to query the Sherlock sky-context system.

19.1 Ways to use the API

The Lasair API uses either HTTP GET or POST. Arguments can be passed in the query string, as JSON or form encoded. Responses are JSON. There is a throttling system in the backend: users with an account get up to 100 calls per hour, but "power" users get up to 10,000 calls per hour. If you wish your account to be upgraded to power user, email Lasair-help

The examples below show how to drive the API with either GET URL, POST curl or python with the 'lasair' package. The URL should be pasted into a web browser. The curl script pasted into a terminal window, and the python code copied into a file and executed as a python program.

19.2 Throttling of API Usage

The Lasair API counts numbers of calls on a per-user basis, and restricts the number that can be executed in any hour time period. There are also restrictions on the number of rows that can be returned by the 'query' method. To use the API with less throttling, please get your own token from your Lasair account, as explained below "Get Your Token". If you would like to use the system for serious work, please email Lasair-help, explain what you are doing, and you will be put into the "Power Users" category. The limits for these three categories of user are:

- User token (see 'Get Your Token') below: 100 API calls per hour, maximum 10,000 rows from query.
- Power user token (on request): 10,000 API calls per hour, maximum 1,000,000 rows from query.

Note: WE ASK YOU TO PLEASE NOT SHARE THESE TOKENS. If you share code that uses the Lasair API, please put the token in a separate, imported file or environment variable, that you do not share, and is not put in github.

19.2.1 Authorisation Token

Request authentication is via the API key. For GET queries, the key can go in the parameter string, and for POST queries, the key goes in the headers. In the following, the string of xxxxxxxxxxxxxx characters should be replaced by your own key.

19.2.2 Get Your Token

Once you are logged in on the website, clik on your name at the top right, and choose “My Profile”. The API key is shown there.

19.3 Methods

Click on the method name to jump to documentation in the reference below.

- */api/cone/*: runs a cone search on all the objects in the Lasair database.
- */api/query/*: runs a SQL SELECT query on the Lasair database.
- */api/streams/*: returns a record of the output from a Lasair streaming query.
- */api/objects/*: returns a machine-readable version of the object web page.
- */api/lightcurves/*: returns simple lightcurves for a number of objects.
- */api/sherlock/objects/*: returns Sherlock information about a list of named objects.
- */api/sherlock/position/*: returns Sherlock information about a sky position.

19.4 /api/cone/

This method runs a cone search on all the objects in the Lasair database. The arguments are:

- `ra`: (float) the right ascension in decimal degrees,
- `dec`: (float) the declination in decimal degrees,
- `radius`: (float) the angular radius of the cone in arcseconds, the maximum being 1000 arcseconds.
- `requestType`: (string) the type of request, which can be:
  - `nearest`: returns only the nearest objects within the cone
  - `all`: returns all the objects within the cone
  - `count`: returns the number of objects within the cone

GET URL Example

```
https://lasair-ztf.lsst.ac.uk/api/cone/?&ra=194.494&dec=48.851&radius=240.0&
↪requestType=all&token=xxxxxxxxxxxxxxxxxxxxxxxxx&format=json
```

Curl Example: The API key (token) goes in the header of the request, and the data in the data section.

```
curl --header "Authorization: Token xxxxxxxxxxxxxxxxxxxxxxxxx" --data "ra=194.494&
↪dec=48.851&radius=240.0&requestType=all" https://lasair-ztf.lsst.ac.uk/api/cone/
```

Python Example: This code requires the `lasair` library.

```
import lasair
token = 'xxxxxxxxxxxxxxxxxxxxxxxxx'
L = lasair.lasair_client(token)
c = L.cone(ra, dec, radius=240.0, requestType='all')
print(c)
```

and the return has object identifiers, and their separations in arcseconds, something like:

```
[
    {
        "object": "ZTF17aaajmtw",
        "separation": 2.393511865261539
    }
]
```

19.5 /api/query/

This method runs a query on the Lasair database. There is an interactive query builder, and a schema description. The arguments are:

- `selected`: (string) the list of attributes to be returned,
- `tables`: (string) the list of tables to be joined,
- `conditions`: (string) the "WHERE" criteria to restrict what is returned
- `limit`: (int) (not required) the maximum number of records to return (default is 1000)
- `offset`: (int) (not required) offset of record number (default is 0)

GET URL Example

```
https://lasair-ztf.lsst.ac.uk/api/query/?selected=objectId%2Cgmag&tables=objects&
↪conditions=gmag%3C12.0&token=xxxxxxxxxxxxxxxxxxxxxxxx&format=json
```

Curl Example: The authorization token goes in the header of the request, and the data in the data section.

```
curl --header "Authorization: Token xxxxxxxxxxxxxxxxxxxxxxxx" --data
↪"selected=objectId,gmag&tables=objects&conditions=gmag<12.0&limit=3" https://lasair-
↪ztf.lsst.ac.uk/api/query/
```

Python Example: This code requires the `lasair` library.

```
import lasair
token = 'xxxxxxxxxxxxxxxxxxxxxxxx'
L = lasair.lasair_client(token)
selected    = 'objectId, gmag'
tables      = 'objects'
conditions  = 'gmag < 12.0'
c = L.query(selected, tables, conditions, limit=10)
print(c)
```

and the return is something like:

```
status= 200
[
  {
    "objectId": "ZTF17aaagaie",
    "gmag": 11.4319
  },
  {
    "objectId": "ZTF18aaadvxy",
    "gmag": 11.8582
```

(continues on next page)

(continued from previous page)

```
  },
.... ]
```

19.6 /api/streams// and /api/streams/

This method returns a record of the output from a Lasair streaming query. It represents an alternative to using a Kafka client to fetch from the Kafka server.

If the `topic` URL is provided (with optional `limit`), the contents of the stream are returned. Alternatively, if the topic is not provided, a `regex` argument may be provided, and a list of matching topic names will be returned. A list of all topics can be obtained with the regex `.*` or by omitting the `regex`.

The arguments are:

- `limit`: (int) (not required) the maximum number of records to return (default 1000)
- `regex`: (str) (not required) an expression used to select from the set of topics (regular expression)

GET URL Example with Regex

```
https://lasair-ztf.lsst.ac.uk/api/streams/?regex=.%2ASN.%2A&
↪token=xxxxxxxxxxxxxxxxxxxxxxxx&format=json
```

Curl Example with Regex

The authorization token goes in the header of the request, and the data in the data section.

```
curl --header "Authorization: Token xxxxxxxxxxxxxxxxxxxxxxxx" --data "regex=.\*SN.\*"␣
↪https://lasair-ztf.lsst.ac.uk/api/streams/
```

and the return is a list of topic names, and a URL to get more information:

```
status= 200
[
  {
    "topic": "2SN-likecandidates",
    "more_info": "https://lasair-ztf.lsst.ac.uk/query/2/"
  },
... ]
```

GET URL Example with Topic:

```
https://lasair-ztf.lsst.ac.uk/api/streams/2SN-likecandidates/?limit=1&
↪token=xxxxxxxxxxxxxxxxxxxxxxxx&format=json
```

Curl Example with Topic: The authorization token goes in the header of the request, and the data in the data section. For more information about this stream, see here.

```
curl --header "Authorization: Token xxxxxxxxxxxxxxxxxxxxxxxx" --data "2SN-
↪likecandidates&limit=1" https://lasair-ztf.lsst.ac.uk/api/streams/
```

and the return is something like:

```
status= 200
[
  {
```

(continues on next page)

(continued from previous page)

```
    "objectId": "ZTF19abrokxg",
    "ramean": 35.82811567,
    "decmean": -27.79242059,
    "mjdmin": 59134.39827550016,
    "mjdmax": 59164.37175930012,
    "magrmin": 18.33,
    "rmag": 19.4124,
    "classification": "NT",
    "score": "Not Near PS1 star",
    "UTC": "2020-11-11 09:08:49"
  }
]
```

Python Example with Topic: This code requires the `lasair` library.

```
import lasair
token = 'xxxxxxxxxxxxxxxxxxxxxxxx'
L = lasair.lasair_client(token)
print(L.streams_topics())
c = L.streams('2SN-likecandidates', limit=10)
```

and the return is as above with the curl example

19.7 /api/objects/

This method returns a machine-readable version of the information on a list of objects, which replicates the information on the object page of the web server. The arguments are:

- `objectIds`: a list of objectIds for which data is wanted

GET URL Example

```
https://lasair-ztf.lsst.ac.uk/api/objects/?objectIds=ZTF18abdphvf,ZTF21aapzzgf&
↪token=xxxxxxxxxxxxxxxxxxxxxxxx&format=json
```

Curl Example: The API key goes in the header of the request, and the data in the data section.

```
curl --header "Authorization: Token xxxxxxxxxxxxxxxxxxxxxxxx" --data
↪"objectIds=ZTF18abdphvf,ZTF21aapzzgf" https://lasair-ztf.lsst.ac.uk/api/objects/
```

Python Example: This code requires the `lasair` library.

```
import lasair
token = 'xxxxxxxxxxxxxxxxxxxxxxxx'
L = lasair.lasair_client(token)
c = L.objects(objectIds)
print(c)
```

and the return something like this:

```
status= 200
[
{"objectId":"ZTF18abdphvf",
"objectData":{
    "ncand":8,
```

(continues on next page)

(continued from previous page)

```
    "ramean":293.54591006249996,
    "decmean":49.429229975,
    "glonmean":81.77021010499132,
    "glatmean":13.829346704017533,
.... }
]
```

The data includes everything on the object page, including the object and candidates, as well as the Sherlock and TNS information. The candidate section has bot detections, that have a `candid` attribute, and the much smaller non-detections (upper limits). Each candidate

has links to the cutout images that are shown on the object web page. A complete example

is shown here.

19.8 /api/lightcurves/

This method returns simple lightcurves for a number of objects. **NOTE:** these are difference magnitudes from a reference source, not apparent magnitudes. See this python code to convert the quantities below to apparent magnitude. Each lightcurve is a sequence of detections, or *candidates*, each of which has the quantities:

- `candid`: the candidate ID for the detection
- `fid`: The filter ID for the detection (1 = g and 2 = r)
- `jd`: Julian Day for the detection
- `magpsf`: The difference magnitude
- `sigmapsf`: the error in the difference magnitude.
- `magnr`: Magnitude of the reference source
- `sigmagnr`: the error in the reference magnitude
- `magzpsci`: Zero-point magnitude of the science image
- `isdiffpos`:set to 't' if positive difference magnitude, 'f' for negative

The arguments are:

- `objectIds`: (string) comma-separated string of objectIds to be fetched
- There is a upper limit on the number of lightcurves that can be fetched, currently 50. If you need to do serious data mining on Lasair light curves, please write to contact the Lasair team.

GET URL Example

```
https://lasair-ztf.lsst.ac.uk/api/lightcurves/?objectIds=ZTF20acgrvqo%2CZTF19acylwtd
↪%2CZTF18acmziob&token=xxxxxxxxxxxxxxxxxxxxxxxxx&format=json
```

Curl Example: The authorization token goes in the header of the request, and the data in the data section.

```
curl --header "Authorization: Token xxxxxxxxxxxxxxxxxxxxxxxxx"
--data "objectIds=ZTF20acgrvqo,ZTF19acylwtd,ZTF18acmziob"
https://lasair-ztf.lsst.ac.uk/api/lightcurves/
```

Python Example: This code requires the `lasair` library.

```
import lasair
token = 'xxxxxxxxxxxxxxxxxxxxxxxx'
L = lasair.lasair_client(token)
c = L.lightcurves(['ZTF20acgrvqo','ZTF19acylwtd','ZTF18acmziob'])
print(c)
```

and the return is something like:

```
{
  "objectId": "ZTF20acpwljl",
  "candidates":[
    {
      "candid":1961223331015015002,
      "fid":1,
      "magpsf":19.46470069885254,
      "sigmapsf":0.12973499298095703,
      "magnr":21.349000930786133,
      "sigmagnr":0.0989999994635582,
      "magzpsci":26.41069984436035,
      "isdiffpos":"t",
      "jd":2459715.7233333,
    },
    ]
```

19.9 /api/sherlock/objects/

This method returns Sherlock information for a collection of named objects, either the "lite" record that is also in the Lasair database, or the full record including many possible crossmatches. The arguments are:

- `objectIds`: a comma-separated list of objectIds, maximum number is 10
- `lite`: Set to 'true' to get the lite information only

GET URL Example with objects

```
https://lasair-ztf.lsst.ac.uk/api/sherlock/objects/?objectIds=ZTF20acpwljl
↪%2CZTF20acqqbkl%2CZTF20acplggt&token=xxxxxxxxxxxxxxxxxxxxxxxx&lite=true&format=json
```

Curl Example with list of objects: The authorization token goes in the header of the request, and the data in the data section.

```
curl --header "Authorization: Token xxxxxxxxxxxxxxxxxxxxxxxx" --data
↪"objectIds=ZTF20acpwljl,ZTF20acqqbkl,ZTF20acplggt&lite=True" https://lasair-ztf.
↪lsst.ac.uk/api/sherlock/objects/
```

Python Example with list of objects: This code requires the `lasair` library.

```
import lasair
token = 'xxxxxxxxxxxxxxxxxxxxxxxx'
L = lasair.lasair_client(token)
c = L.sherlock_objects(['ZTF20acgrvqo','ZTF19acylwtd','ZTF18acmziob'])
print(c)
```

and the return is something like:

```
{
    "ZTF20acpwljl": {
        "classifications": {
            "ZTF20acpwljl": [
                "SN",
                "The transient is possibly associated with <em><a href='http://
↪skyserver.sdss.org/dr12/en/tools/explore/Summary.aspx?id=1237673709862061782'>SDSS␣
↪J081931-060114.9</a></em>; a J=17.01 mag galaxy found in the SDSS/2MASS/PS1␣
↪catalogues. It's located 1.09 arcsec N, 1.11 arcsec W from the galaxy centre."
            ]
        },
        "crossmatches": [
            {
                "catalogue_object_id": "1237673709862061782",
                "J": 17.007,
                "JErr": 0.215,
                "H": 15.974,
                "HErr": 0.179,
                "K": 15.389,
```

19.10 /api/sherlock/position/

This method returns Sherlock information for an arbitrary position in the sky, either the "lite" record that is also in the Lasair database, or the full record including many possible crossmatches. It is meant as an illustration of what Sherlock can do. If you would like to use Sherlock for high volume work, please Email Lasair-help. The arguments are:

- `ra`: Right ascension of a point in the sky in degrees
- `dec`: Declination of a point in the sky in degrees
- `lite`: Set to 'true' to get the lite information only

GET URL Example

```
https://lasair-ztf.lsst.ac.uk/api/sherlock/position/?ra=16.851866&dec=34.53307&
↪token=xxxxxxxxxxxxxxxxxxxxxxxx&format=json
```

Curl Example: The authorization token goes in the header of the request, and the data in the data section.

```
curl --header "Authorization: Token xxxxxxxxxxxxxxxxxxxxxxxx" --data "ra=16.851866&
↪dec=34.53307" https://lasair-ztf.lsst.ac.uk/api/sherlock/position/
```

Python Example: This code requires the `lasair` library.

```
import lasair
token = 'xxxxxxxxxxxxxxxxxxxxxxxx'
L = lasair.lasair_client(token)
c = L.sherlock_position(['ZTF20acgrvqo','ZTF19acylwtd','ZTF18acmziob'])
print(c)
```

and the return is something like:

```
status= 200
{
  "classifications": {
```

(continues on next page)

(continued from previous page)

```
    "query": [
      "VS",
      "The transient is synonymous with
```

CHAPTER

TWENTY

PYTHON NOTEBOOKS

Table of Contents

Below are some Google colab notebooks that use Lasair API, which has a token authentication scheme.

The notebooks below have a fake token “xxxxxxx” that doesn’t work. To use the notebooks, you will need to :

- Get a Lasair account here, and log in to the Lasair website.
- Click on your username at the top right and select “My Profile”
- Copy the notebook to your own Google account, and replace the xxxxxxx with your own token.

This notebook does not require an API key.

Please Contact us with any notebooks that you would like to share.

CHAPTER

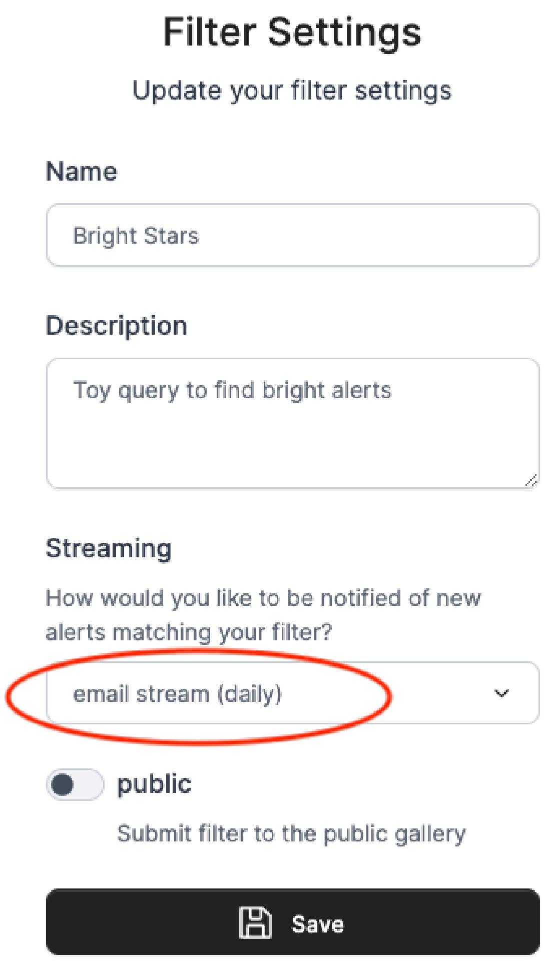
TWENTYONE

ALERT STREAMS

The Lasair broker can send immediate "push" notifications when your active query/filter sees and interesting alert. Here is how to make that happen with email notification. First make sure you are logged in to your Lasair account (top left of screen, then go to create new stored query. This page is about how to get email alerts from your active query; the process is very similar for Kafka alerts, except that you will fetch the results by machine instead of by email. See article Reading a Kafka Stream.

You will need to be logged in to your Lasair account. Make a filter as in the

previous section, then click "Save". You will then be prompted for "Filter Settings", which you can fill in like this:



Filter Settings

Update your filter settings

Name: Bright Stars

Description: Toy query to find bright alerts

Streaming

How would you like to be notified of new alerts matching your filter?

email stream (daily)

public

Submit filter to the public gallery

Nothing will happen immediately. You can run the query in the usual way from the web browser, but you will have to wait for some alerts to arrive before your active query will be triggered. Once that happens, you will get an email at the address you used to create your account. Something like the message shown here. Note that the attributes you chose above are reported (objectId, gmag, since), together with the UTC time at which the alert was triggered.

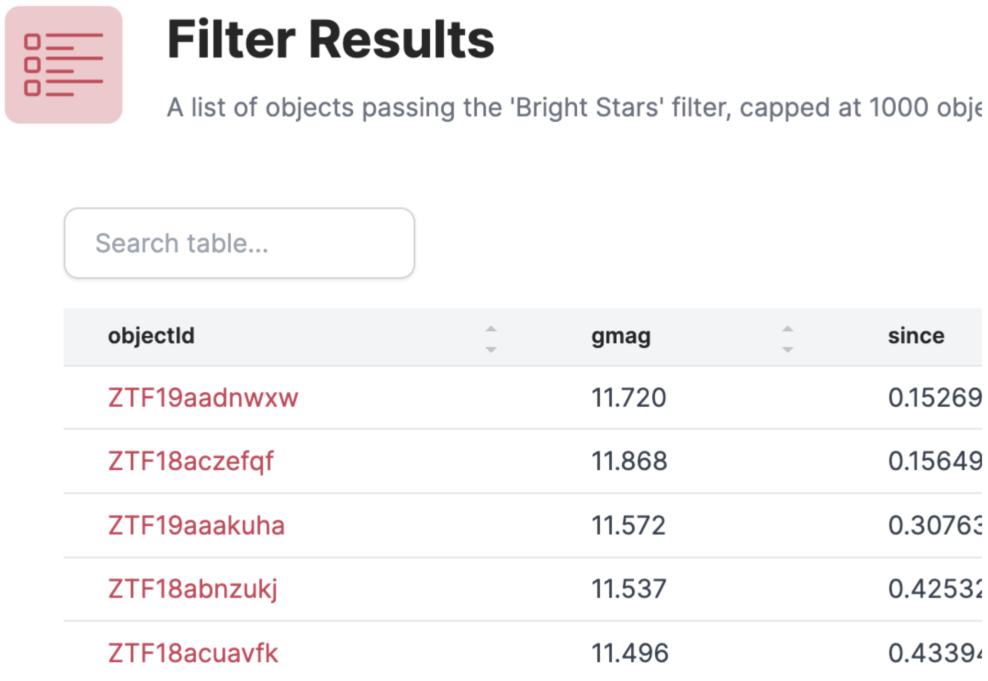
The email distribution is intended for filters that do not pass many alerts

in a given day, or else the email box will be flooded with spam. Lasair throttles

the number of emails; once the first has been sent, another willnot be sent until 24

hours later, containing the objects passed by the filter in that time. In this

way, a maximum of one email per day can come from a Lasair filter.


Filter Results

A list of objects passing the 'Bright Stars' filter, capped at 1000 obj

objectId	gmag	since
ZTF19aadnwxw	11.720	0.15269
ZTF18aczefqf	11.868	0.15649
ZTF19aaakuha	11.572	0.30763
ZTF18abnzukj	11.537	0.42532
ZTF18acuavfk	11.496	0.43394

21.1 Kafka Streams

While there are many methods to handle emails automatically, they are complex and beset with problems.

Therefore Lasair also provides a protocol for immediate delivery that is more suitable for machines to communicate with machines. It is called Kafka.

By providing Kafka streams, Lasair provides a machine-readable packet of data that can cause action at your site.

While this can be done with the email channel, it is awkward. To convert your filter from email to kafka, bring

choose the filter, then click the Settings button, and change the streaming option to 'kafka stream'.

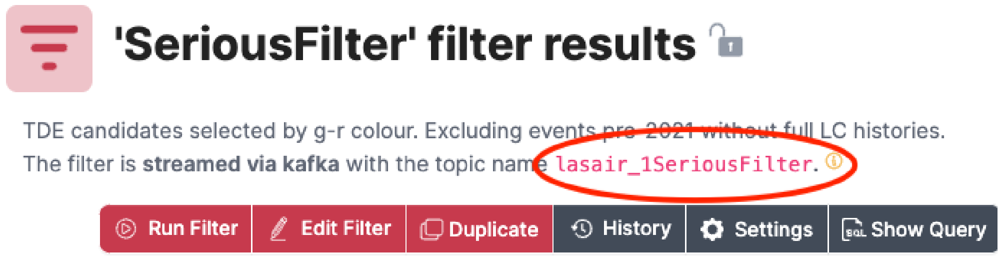
- We recommend Confluent Kafka, the python install being `pip install confluent_kafka`.
- You will be connecting to kafka.lsst.ac.uk on port 9092

You will need to understand two concepts: Topic and GroupID.

- The Topic is a string to identify which stream of alerts you want, which derives from the name of a Lasair streaming query.

- The GroupID tells Kafka where to start delivery to you. It is just a string that you can make up, for example "Susan3456". The Kafka server remembers which GroupIds it has seen before, and which was the last alert it delivered. When you start your code again with the same GroupID, you only get alerts that arrived since last time you used that GroupId. If you use a new GroupID, you get the alerts from the start of the Kafka cache, which is about 7 days.

You can find the topic that corresponds to your filter in the detail page, shown here in the red oval:


'SeriousFilter' filter results

TDE candidates selected by g-r colour. Excluding events pre-2021 without full LC histories.
The filter is **streamed via kafka** with the topic name lasair_1SeriousFilter.

The topic name is a combination of the string "lasair_", the ID number of your user account, and

a sanitised version of the name you gave the filter. Therefore if you edit the filter and change its name,

the topic name will also change.

For testing purposes, the `group_id` will change frequently, and you can get all of the alerts

the come from the given stream. Then you will set up your program to run continuously,

perhaps in a `screen` session on a server machine, or started every hour by `cron`.

In this case, the `group_id` should remain constant, so you won't get any alerts twice.

Here is the sample code

```
import json
from lasair import lasair_consumer

kafka_server = 'kafka.lsst.ac.uk:9092'
group_id     = 'test123'
my_topic     = 'lasair_2SN-likecandidates'
consumer = lasair_consumer(kafka_server, group_id, my_topic)
import json
n = 0
while n < 10:
    msg = consumer.poll(timeout=20)
    if msg is None:
        break
    if msg.error():
        print(str(msg.error()))
        break
    jmsg = json.loads(msg.value())
    print(json.dumps(jmsg, indent=2))
    n += 1
print('No more messages available')
```

CHAPTER

TWENTYTWO

MAKING AN ANNOTATOR

22.1 Request to Lasair team

Annotation means that external users push information to the Lasair database.

Therefore it requires that user to inform the Lasair team and be approved

before it will work. The team will use the admin interface to create an `annotator`

object in the database, which is a conneciton between the API token of that user

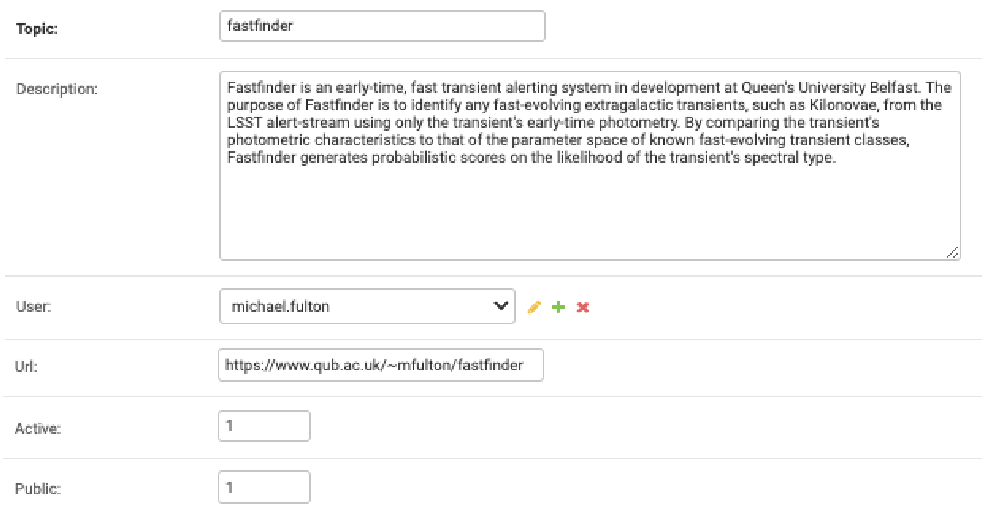
with the name (`topic`) assigned to the annotator.

In the first case, write to Lasair Team

to propose your annotator.

Here we see that admin interface that the Lasair team uses to make your annotator,

if it is approved.:



Topic:	fastfinder
Description:	Fastfinder is an early-time, fast transient alerting system in development at Queen's University Belfast. The purpose of Fastfinder is to identify any fast-evolving extragalactic transients, such as Kilonovae, from the LSST alert-stream using only the transient's early-time photometry. By comparing the transient's photometric characteristics to that of the parameter space of known fast-evolving transient classes, Fastfinder generates probabilistic scores on the likelihood of the transient's spectral type.
User:	michael.fulton
Url:	https://www.qub.ac.uk/~mfulton/fastfinder
Active:	1
Public:	1

You can see that your new annotator is a name (or *topic*) and description.

It is bound to your own Lasair account, meaning that you must use your own API token

to run your annotator code (see below). There is also a URL for further information.

If the annotator is `active`, it can receive annotations, meaning that if the external system

goes wild, the annotator can be switched off by setting `active=0`. Finally, the annotator can be `public` or not, meaning that it is visible or not to others building Lasair filters.

22.2 Make the code

The following code reads the stream from a Lasair filter, and for each `objectId`,

it pulls the complete object information so it could analyse the lightcurve

and other information before making a classification decision.

This information is collected up as the annotation and sent back to Lasair,

where it will be available for others to query.

There should be a file `settings.py` file to accomany the code below with these variables defined::

- `TOPIC_IN`: The name of your streaming query as seen in the filter detail, where it says "The filter is streamed via kafka with the topic name"
- `GROUP_ID`: Choose a new one evey run when testing; keep it constant for long-term running
- `API_TOKEN`: As found in 'My Profile' top right of the web page
- `TOPIC_OUT`: The name of your annotator as agreed with the Lasair team (above)

If the code below is not clear, it would be good for you to read about how

the (Lasair client)[rest-api.html] works.

For more information about what is returned as `objectInfo`, a complete example

is shown here.

For testing purposes, the `GROUP_ID` will change frequently, and you get all of the alerts

the come from the given stream. Then you will set up your annotator program to run continuously,

perhaps in a `screen` session on a server machine, or started every hour by `cron`.

In that case, the `GROUP_ID` will remain constant, so you won't get any alerts twice.

A much simpler code is possible if for example the annotation is the classification

results from another broker. In that case, only the call to `L.annotator()` is necessary.

```
import json, sys, settings
import lasair

# This function deals with an object once it is received from Lasair
def handle_object(objectId, L, topic_out):
    # from the objectId, we can get all the info that Lasair has
    objectInfo = L.objects([objectId])[0]
    print(objectInfo.keys())
    if not objectInfo:
        return 0
    # objectInfo.keys():
    #  -- objectData: about the object and its features
    #  -- candidates: the lightcurve of detections and nondetections
    #  -- sherlock: the sherlock information
    #  -- TNS: any crossmatch with the TNS database

    # analyse object here. The following is a toy annotation
    classdict      = {'fruit': 'apple'}
```

(continues on next page)

(continued from previous page)

```
    classification = 'ripe'
    explanation    = 'another nice apple'

    # now we annotate the Lasair data with the classification
    L.annotate(
        topic_out,
        objectId,
        classification,
        version='0.1',
        explanation=explanation,
        classdict=classdict,
        url='')
    print(objectId, '-- annotated!')
    return 1

#####################################
# first we set up pulling the stream from Lasair
# a fresh group_id gets all, an old group_id starts where it left off
group_id = settings.GROUP_ID

# a filter from Lasair, example 'lasair_2SN-likecandidates'
topic_in = settings.TOPIC_IN

# kafka consumer that we can suck from
consumer = lasair.lasair_consumer('kafka.lsst.ac.uk:9092', group_id, topic_in)

# the lasair client will be used for pulling all the info about the object
# and for annotating it
L = lasair.lasair_client(settings.API_TOKEN)

# TOPIC_OUT is an annotator owned by a user. API_TOKEN must be that users token.
topic_out = settings.TOPIC_OUT

# just get a few to start
max_alert = 5

n_alert = n_annotate = 0
while n_alert < max_alert:
    msg = consumer.poll(timeout=20)
    if msg is None:
        break
    if msg.error():
        print(str(msg.error()))
        break
    jsonmsg = json.loads(msg.value())
    objectId       = jsonmsg['objectId']
    n_alert += 1
    n_annotate += handle_object(objectId, L, topic_out)

print('Annotated %d of %d objects' % (n_annotate, n_alert))
```

CHAPTER

TWENTYTHREE

LASAIR ACCOUNT

We encourage users to signup for a Lasair account. All you need is a valid email address.

You will then be able to store queries and convert them to filters,

build watchlists and watchmaps, and use the Lasair API and client.

Click here to sign up for an account

CHAPTER TWENTYFOUR

QUESTIONS AND ANSWERS

24.1 What are Lasair-ZTF and Lasair-LSST?

See ZTF and LSST

24.2 What can I get from this web site?

The Lasair alert broker gives access to millions of astronomical transient detections: when a star or galaxy becomes brighter or fainter than it was at an earlier time.

24.3 What data does Lasair offer?

Whenever a star or galaxy in the sky changes brightness, it is given an "objectId", which can be used to see all the data about that object. Data includes a "light curve" of brightness measurments at different times, in different filters; crossmatching with existing source catalogs, and other data.

Changes in brightness are transferred to the Lasair databases, and pushed to users, within an hour

of the telescope taking the observation.

24.4 What is here for an amateur astronomer?

A serious amateur telescope would have a 500 mm aperture, with a limiting magnitude of about 16, costing over $40,000. In any year there will be a few supenovae visible to this system.

24.5 How can I ask a question to the Lasair team?

Write to the help email: lasair-help at lists.lasair.roe.ac.uk.

24.6 How can I use my knowledge of SQL to use Lasair?

Each filter in Lasair is an SQL SELECT query. The syntax is "SELECT <attributes> FROM <tables> WHERE <conditions;>" The attributes come from the schema – shown to the right in the filter builder page. The tables are selected from `objects`, `sherlock_classifications`, `crossmatch_tns`, as well as any watchlists, watchmaps or annotations you choose. The conditions in the WHERE clause allow a simplified SQL, using just comparison operators, without operators such as "group" and "having".

24.7 How can I query the Lasair database?

You can type SQL into the filter builder – instructions here,

and you can run a query somebody else has made that is public. If you sign up and login to Lasair, you can save your queries and you can copy somebody else's query then modify it.

24.8 What is the difference between a Query and a Filter?

A query operates on the whole database of alerts, but a filter only runs on new alerts, as they stream from the telescope. They are very similar ideas: but query implies running on the database of past alerts,

and filter implies running on the stream of incoming new alerts.

24.9 What is the schema of the Lasair database?

Can be found at the schema page.

24.10 How do I choose which alerts are interesting to me?

Choosing interesting alerts can be based on several criteria: The characteristics of the light curve; coinicdence of the alert with a galaxy or known variable star; coincidence of the alert with one of the sources in which you are interested (a watchlist); location of the alert in a given area of the sky, for example a gravitational wave skymap.

24.11 Why should I register on the Lasair website?

Registration is easy, and just requires a valid email (signup here). You can then build and save queries, watchlists, and watchmaps (sky areas), convert those to real-time slert treams, and use the Lasair API.

24.12 Besides Lasair, what other websites carry astronommical transients?

There are seven community brokers that will receive and process LSST alerts in real time: ALeRCE, AMPEL, ANTARES, BABAMUL, Fink, Lasair, and Pitt-Google.

24.13 How long has Lasair been operating?

Lasair has been processing, storing, and distributing alerts from the ZTF survey since 2018.

Operation with LSST will start in 2023.

24.14 Why are there no alerts on the Lasair front page?

The front page shows alerts from the last seven days. Sometimes no alerts have been received

in that time, and so none are shown. Reasons may be weather or equipment failure.

More information is available in the green news bar at the top of the front page.

24.15 Can I get alerts from a particular region of the sky?

Lasair supports "watchmaps", defined by a MOC, that you build yourself.

24.16 Can I get alerts associated with my favourite sources?

You can build a "watchlist" of your favourite sources, and build a corresponding query that includes crossmatch with that watchlist. Instructions are here.

24.17 Can Lasair alert me about supernovae and kilonovae?

There are some filters already built that find alerts in the outskirts of galaxies. There are also queries that find supernovae already reported to the Transient Name Service.

24.18 Can Lasair alert me about gravitational-wave events?

Not yet, but soon.

24.19 How can I find out about the LSST survey and the Vera Rubin Observatory?

General FAQ on LSST and Rubin is here, about community alert brokers in particular here

24.20 How can I write code and notebooks that use the Lasair database?

The Lasair client is described here, and

there are sample notebooks here.

24.21 Does Lasair classify alerts into classes?

Lasair supports the idea of *annotation*, where external users and other brokers build and share classification information with Lasair. These annotations can then be used as part of Lasair filters.

24.22 Does Lasair have an API?

The Lasair client is described here.

24.23 What is difference magnitude compared to apparent magnitude?

This is explained here.

24.24 How do search for an object by position in the sky?

This is called a "cone search". See next question.

24.25 What is a cone-search and can Lasair do this?

A *cone* in this context means a point in the sky with an angular tolerance – the opening angle of the cone, as explained here.

You can use the Lasair Sky Search

to do this.

24.26 How can I do 1000 cone searches all at once?

The efficient way to do this is to build a watchlist,

as explained here. If the watchlist has

less than 10,000 sources, there is button on the watchlist page to crossmatch

with all past objects.

24.27 Can I see sky images in different wavelengths around a Lasair alert?

The Lasair object page has a panel of AladinLite

that shows many kinds of sky image, from radio to gamma, and can be zoomed in and out.

24.28 When I make a filter, can I share it with my colleagues?

Filters, watchlists, and watchmaps can all be made public so that others can see them.

A public filter can be copied and modified.

24.29 Can I get immediate notification of interesting alerts?

See the section on alert streams.

CHAPTER TWENTYFIVE

CONTACT US

We really value feedback, especially from our users.

Please ask your question or report your bug with the Rubin Community forum under Support > Lasair.

You will need to get an account there – a simple process, give your email and respond to a message sent there.

25.1 The Lasair Team

- University of Edinburgh
  - Gareth Francis
  - Andy Lawrence
  - Terry Sloan
  - Roy Williams
- Queen's University Belfast
  - Stephen Smartt
  - Ken Smith
  - Dave Young

CHAPTER

TWENTYSIX

LASAIR INGESTION STATUS

To find out about the current status of Lasair ingestion

CHAPTER

TWENTYSEVEN

ACKNOWLEDGEMENTS

Lasair is supported by the UKRI Science and Technology Facilities Council and is a collaboration between the University of Edinburgh (grant ST/N002512/1) and Queen's University Belfast (grant ST/N002520/1) within the LSST:UK Science Consortium. ZTF is supported by National Science Foundation grant AST-1440341 and a collaboration including Caltech, IPAC, the Weizmann Institute for Science, the Oskar Klein Center at Stockholm University, the University of Maryland, the University of Washington, Deutsches Elektronen-Synchrotron and Humboldt University, Los Alamos National Laboratories, the TANGO Consortium of Taiwan, the University of Wisconsin at Milwaukee, and Lawrence Berkeley National Laboratories. Operations are conducted by COO, IPAC, and UW. This research has made use of ``Aladin sky atlas'' developed at CDS, Strasbourg Observatory, France 2000A\&AS..143…33B and 2014ASPC..485..277B.