Contrib. Astron. Obs. Skalnaté Pleso **56/1**, 186 – 193, (2026)
https://doi.org/10.31577/caosp.2026.56.1.186

Alertissimo - a tool for orchestration of LSST broker streams

V. Vujčić[1], V.A. Srećković[2] and S. Babarogić[3]

[1] *Astronomical Observatory, Volgina 7, 11060 Belgrade, Serbia (E-mail: veljko@aob.rs)*

[2] *University of Belgrade, Institute of Physics Belgrade, PO Box 57, 11001 Belgrade, Serbia*

[3] *University of Belgrade, Faculty of Organizational Sciences, Jove Ilića, 11000 Belgrade, Serbia*

Received: September 25, 2025; Accepted: November 19, 2025

Abstract. The Vera C. Rubin Observatory, through its Legacy Survey of Space and Time, will soon start producing 10 million alerts on transient astronomical objects per night. Due to logistics and bandwidth, alerts will not be dispatched directly to the public but to 'brokers' i.e. tools selected by LSST to handle alert streams. Brokers offer both common, specific and micro-specific functionalities related to alert handling, analysis, representation and dissemination. In this ecosystem, potentially augmented by data streams from other astronomical sources, there is a - need demonstrated by the community - for use cases which combine features of individual brokers. In this paper we present initial efforts and a prototype of such a tool, along with a language that would allow users to define use cases / workflows in a manner tailored for the domain.

Key words: astronomical transients – large astronomical surveys – Vera C. Rubin Observatory – real time event processing – domain-specific languages – natural language processing

1. Introduction

As a new era of astronomical data science is about to kick off with Vera C. Rubin Observatory's Legacy Survey of Space and Time (Rubin/LSST) Ivezić et al. (2019) production phase, the ecosystem of tools and infrastructure dealing with transient astronomical events looks fit and (almost) ready for the data burst. It is commonly known that the LSST Alert Production pipeline (Bosch et al., 2018) will capture and process images, perform difference-image analysis and distribute alerts of every astronomical source with a signal-to-noise ratio (SNR) ratio > 5 in positive or negative flux (Graham et al., 2019a). Due to bandwidth and other considerations, alerts will not be directly available to the public but through 'brokers' - tools developed by different scientific teams worldwide, specifically made for and approved by LSST. At least six brokers (AleRCE, AMPEL, ANTARES, Fink, Lasair, Pitt-Google) will be able to ingest LSST alert

stream in near-real time, analyze, classify, offer UI tools, programming APIs and more. Some of the main features of brokers overlap but they differ significantly in how they expose these features, with even greater divergence in their underlying implementations and backend technologies. (Vujčić et al., 2025).

2. Alertissimo

We introduce Alertissimo (2025) - a tool for orchestration of broker streams and potentially for a wider scope of astro-science use cases. It can be seen as a novel topic but also an extension of the work done and analyzed by the Serbian group related to astronomical stream processing and virtual observatory (Vujčić & Jevremović, 2020; Jevremović et al., 2020). This tool is conceived on two premises:
a) that there is a need for building workflows out of multiple LSST broker data streams (and potentially external sources as well);
b) that there is a need for a tool that would allow expressing such workflows in a powerful yet approachable manner.
We can roughly name these two premises as 'Broker Orchestration' and 'DSL for Transients'. We also present Alertissimo's overall architecture.

2.1. Broker orchestration

As stated above, at least 6 LSST brokers are in a mature development phase, and some of them - like Lasair, ALeRCE, ANTARES and Fink - have worked in production with Zwicky Transient Facility (ZTF) data (Graham et al. 2019b, data volume order of magnitude lower than LSST). Apart from technological choices, implementation differences and performance, brokers also vary in features (and microfeatures), and in the ways they are exposed through their UIs and APIs. There are ongoing discussions in the Transients and Variable Stars scientific collaboration (LSST-TVS) on use cases that combine features of various brokers (see the Appendix A for a use case featuring supermassive binary black hole (SMBBH) detection). There may also be availability and responsiveness considerations, in a technical sense.

Brokers offer external access to their features through APIs, most through REST[1] and Python, some through a Python interface only. Here we have a similar but non-standardized set of their exposed methods, not only related to major feature differences but also to naming conventions and microfeatures (e.g - how do they retrieve single vs multiple objects? can they handle SQL? how does output look like and can you control granulation? etc). API access control is implemented differently across brokers, typically requiring users to supply

[1]Representational State Transfer, an architectural style for web services which defines GET and POST methods for data retrieval

credentials - such as tokens or username/password combinations - with varying permission models and granularity.

It is a relatively easy task to write a program or a script that would pick specific methods from various brokers and perform a single use case. Alertissimo should act here as a generator of scientific scenarios - take any number of features from any broker (or other available source) and orchestrate them with one another. The primary function of Alertissimo here is one of a science-enabler, where all decisions are in control of the user[2].

Alertissimo is designed with an input-agnostic core, where user-specified workflows are first translated into intermediate representations (IR) with a well-defined structure built using Pydantic[3] models. These IR models are the product of a conceptual analysis of broker features; their design is generalized, while their implementation is specialized through a polymorphic class structure.

2.2. DSL for transients

The challenge of translating high-level user specifications into executable workflows is often addressed through Domain-Specific Languages (DSLs) and workflow systems (Gil et al., 2010). There are three main directions for Alertissimo UI development, one of them being the backbone of the others - the DSL for Transients. The two others, Natural Language Processing (NLP) Chowdhary (2020) and visual workflow builder are planned to be built once a mature DSL foundation is established. DSLs (Mernik et al., 2005) are programming languages designed for a very specific purpose and are usually of declarative nature[4]. Alertissimo's DSL for Transients acts as the semantic and syntactic backbone linking user intent with system capabilities. It exposes a structured, declarative representation of broker operations, allowing every workflow to be expressed in a consistent formalism. This provides that all interactions, no matter how informal their starting point was, ultimately resolve into a precise and testable specification. The demo version of Alertissimo featuring DSL for Transients is currently available through a github branch (`https://github.com/sambolino/alertissimo/tree/feature/dsl`). Example of DSL for Transients is found in the Appendix A.

The natural language interface in Alertissimo will enable users to describe their scientific intentions in plain terms, while maintaining expressive power.

[2]However, the idea persists that Alertissimo could also act in a 'light' AI-based counseling manner too, based on the knowledge of previous user experiences/success rates. This kind of development could take place in later phases.

[3]Pydantic is a Python library designed for data validation, parsing, and serialization which provides a structured way to define data schemas and ensures that incoming data conform to the specified types and constraints.

[4]Declarative programming languages are stating 'what' should be done in contrast with imperative languages which are describing 'how' it should be done. All general programming languages are imperative while more narrowly specified languages, such as SQL, rule-based languages, HTML etc are declarative.

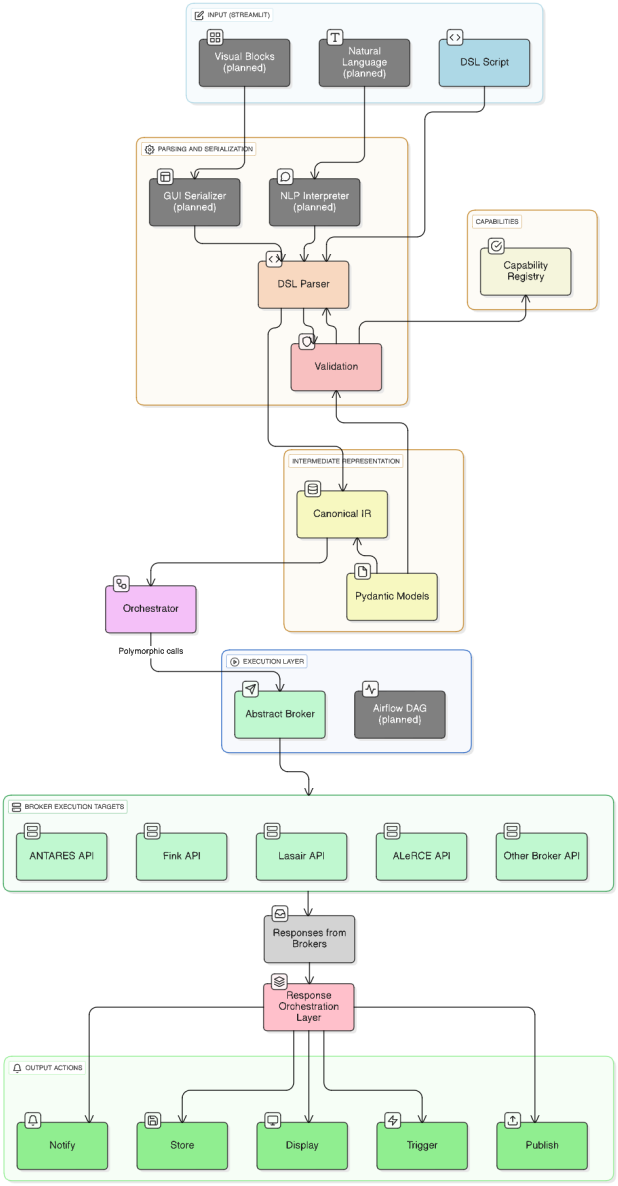

Figure 1. End to end flow diagram of Alertissimo modules illustrating the pipeline from DSL input through orchestration to broker execution. For detailed description see Subsection 2.3

The aim is to test both a pure NLP prompt, where users would describe use-cases in written English, and a chat-bot style AI dialogue which could act as an assistant providing refinement and disambiguation, even for users initially unfamiliar with broker architectures. Both the natural language and visual interfaces of Alertissimo will be developed on top of a formally defined DSL grammar. As DSL defines the valid constructs, relationships, and parameter types for workflows involving broker orchestration, it also ensures that any interface - textual, conversational, or visual - ultimately produces valid, executable specifications. This formal grammar serves as the common translation layer: natural language inputs are parsed and normalized into DSL expressions, and also the visual interface directly maps user manipulations of blocks and links into equivalent DSL statements.

From a broader perspective, this layered UI approach reflects a continuum between accessibility and expressiveness. At one end, the conversational interface empowers domain scientists to describe ideas without syntactic constraints; at the other, the DSL offers power users fine-grained control and versionable scripts suitable for integration into research pipelines. The visual interface bridges these modes, exposing the structure of the DSL while remaining approachable.

2.3. Architecture overview

The overall architecture of Alertissimo is depicted in Figure 1, with data flow and key components as follows:

1. Input Layer: Users can define their use-cases via multiple interfaces (DSL, NLP/chatbot (planned), Visual blocks - Yahoo Pipes (Pruett, 2007) style (planned)), with the DSL being the currently implemented method.

2. Parsing and Validation: User input is reduced to DSL formalism to the appropriate parser (DSL Parser, etc.). Validation is performed both in terms of grammar and broker capabilities.

3. Capabilities: Broker capabilities are defined within yaml[5] files

4. IR Generation: Valid DSL is being translated into IR models, still being execution-agnostic.

5. Orchestration and Execution: The validated IR is processed by the Orchestrator. This component manages the workflow by making polymorphic calls to the Execution Layer, which consists of an Abstract Broker interface. This design allows the system to specialize the implementation for different targets

[5]YAML is a human-readable data serialization language primarily used for configuration files and data storage

while maintaining a generic core. For the sake of performance/execution control, The orchestrator can also trigger a workflow coordination framework, such as an Airflow DAG (planned).

6. Response Handling and Output: Responses from brokers are collected and processed by and the system executes one or more output actions, such as to Notify a user, Store the data, Display it, Trigger a downstream process, or Publish the results.

3. Conclusion

Alertissimo is a new tool for building scientific workflows out of streams of alerts generated by Rubin/LSST. The alerts are not disseminated directly to the public, but through tools called 'brokers' - projects developed for ingestion and analysis of LSST transient alerts. Alertissimo covers all of the brokers' concepts and features and offers an integrated interface for orchestrating multiple brokers within a single workflow. On top of Alertissimo lies a domain-specific language devised to easily yet expressively represent scientific use cases. NLP/LLM/chatbot and visual workflows are planned as extensions that dependably translate and validate to the canonical DSL.

Together, these interaction paradigms establish a scalable human–machine interface strategy, ensuring that Alertissimo remains adaptable as new brokers, data modalities, and analysis paradigms emerge in astronomy and astrophysics. Our prototype demonstrates how broker interoperability and workflow definition can empower the astronomy and astrophysics community in the era of Rubin/LSST.

Acknowledgements. This research was supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia (MSTDIRS) through contract no. 451-03-66/2024-03/200002 made with Astronomical Observatory (Belgrade), 451-03-47/2023-01/200024 made with Institute of physics Belgrade. The authors acknowledge the networking opportunities from the COST Action CA22133 - The birth of solar systems (PLANETS) supported by COST (European Cooperation in Science and Technology).

References

Alertissimo. 2025, Alertissimo, https://github.com/sambolino/alertissimo, accessed: 2025-10-29

Bosch, J., AlSayyad, Y., Armstrong, R., et al., An overview of the LSST image processing pipelines. 2018, *arXiv preprint arXiv:1812.03248*

Chowdhary, K., Natural language processing. 2020, *Fundamentals of artificial intelligence*, 603

Gil, Y., Ratnakar, V., Kim, J., et al., Wings: Intelligent workflow-based design of computational experiments. 2010, *IEEE Intelligent Systems*, **26**, 62

Graham, M., Bellm, E., Guy, L., Slater, C., & Dubois-Felsmann, G. 2019a, *LSST Alerts: Key Numbers* (DMTN-102, URL https://dmtn-102. lsst. io, LSST Data Management Technical Note)

Graham, M., Kulkarni, S., Bellm, E., et al., The Zwicky Transient Facility: Science Objectives. 2019b, *Publications of the Astronomical Society of the Pacific*, **131**, 078001, DOI:10.1088/1538-3873/ab006c

Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al., LSST: From Science Drivers to Reference Design and Anticipated Data Products. 2019, *Astrophysical Journal*, **873**, 111, DOI:10.3847/1538-4357/ab042c

Jevremović, D., Srećković, V., Marinković, B., & Vujčić, V., Databases for collisional and radiative processes in small molecules needed for spectroscopy use in astrophysics. 2020, *Contributions of the Astronomical Observatory Skalnate Pleso*, **50**, 44

Komossa, S., Grupe, D., Marziani, P., et al., The extremes of AGN variability: outbursts, deep fades, changing looks, exceptional spectral states, and semi-periodicities. 2026, *Advances in Space Research*, **77**, 4041

Kovačević, A. B., Popović, L. Č., & Ilić, D., Two-dimensional correlation analysis of periodicity in active galactic nuclei time series. 2020, *Open astronomy*, **29**, 51

Mernik, M., Heering, J., & Sloane, A. M., When and how to develop domain-specific languages. 2005, *ACM computing surveys (CSUR)*, **37**, 316

Pruett, M. 2007, *Yahoo! pipes* (O'Reilly)

Vujčić, V. & Jevremović, D., Real-time stream processing in astronomy. 2020, in *Knowledge Discovery in Big Data from Astronomy and Earth Observation* (Elsevier), 173–182

Vujčić, V., Srećković, V., Babarogić, S., & Aleksić, J., An overview of astronomical transient brokers in Rubin era. 2025, *Contrib. Astron. Obs. Skalnaté Pleso*, **55**, 95

A. DSL for Transients - example code

Here is a sample code of DSL for Transients for supermassive binary black hole (SMBBH) detection, based on the use case presented on the online meeting of the LSST-TVS collaboration. The use case was proposed by prof dr Andjelka Kovačević based on longstanding research of AGN and SMBBH variability (Kovačević et al. (2020), Komossa et al. (2026)). The meeting took place on the Feb 21st 2025. Comments include original definition of the use case from the discussion.

```
# Fink+ALeRCE+Lasair+ANTARES agree on alert
# ('required' argument is added for showcase)
confirm object_id="ZTF25aazqavg" brokers=[fink, alerce, lasair,
    antares] required=3
# ALeRCE retrieves ZTF curves of alerted object
lightcurve broker=alerce survey="ztf"
# Lasair retrieves historical light curves from Pan-STARR of
# alerted objects, and crossmatches with IR, R, X catalogues.
# Helps identify multi-wavelength periodicity.
crossmatch broker=lasair catalog="panstarrs" filters=["ir", "radio"
    , "xray"]
# ANTARES crossmatches alerted object with eROSITA etc for
# further confirmation of multiwavelength of periodicity origin.
crossmatch broker=antares catalog="erosita"
# Lasair provides a Kafka-based alert stream that updates in
# real-time whenever a new observation is made for a
# monitored object.
monitor broker=lasair stream="kafka"
classify method="periodicity_detection"
notify team
store db
```