ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-THPD058

IC@MS – WEB-BASED ALARM MANAGEMENT SOFTWARE*

K. Fugiel, M. Gandor, Sz. Kupiecki, T. Madej, P. Moćko, L. Zytniak†
S2Innovation, Krakow, Poland

Abstract

Modern large-scale scientific facilities, such as particle accelerators and observatories, demand robust alarm management to ensure safe and efficient operations. According to the IEC 62682 industrial standard, the primary role of an alarm system is to promptly notify operators of abnormal conditions or equipment malfunctions and support an effective response. In practice, a control system framework like Tango Controls provides a foundation for building such alarm systems, offering tools for device communication, data archiving, and alarm interfacing in distributed environments.

EXISTING SOLUTIONS

Within the Tango Controls community, multiple alarm-handling solutions have evolved over time [1]. Notably, **Tango Alarm Handler** (developed at Elettra) is a centralized C++ service that evaluates user-defined alarm formulas (logical conditions on device attributes) and uses events to log or announce alarms [2]. In contrast, ALBA's **PANIC** alarm system (built around the PyAlarm Tango device server) employs a distributed approach, running Python-based alarm evaluation rules on multiple servers for flexibility and scalability [3]. These tools (and others, such as an alarm logging database from Soleil) each address alarm management from different angles. However, the lack of a unified interface and the need for standardization (as suggested by IEC 62682) became apparent as facilities grew more complex and collaborative [4].

IC@MS

IC@MS (Integrated Critical Alarms Management Software) was conceived to bridge this gap by providing a unified, modern alarm management platform. Initially developed for Tango-based installations, IC@MS is a modular web-based system that integrates seamlessly with existing Tango alarm infrastructure including both, the PyAlarm devices from ALBA's PANIC and the Alarm Handler service from Elettra – as well as Tango's archiving and database tools [5]. In essence, IC@MS acts as a layer on top of the proven Tango alarm engines, consolidating their outputs and adding advanced user interface, notification, and analysis capabilities. This approach allows facilities to continue using their well-tested alarm generation mechanisms while gaining a centralized alarm console and management hub. Importantly, IC@MS was designed with generality in mind: its modular architecture and technology stack make it adaptable to different sites and even to other control system frameworks with minimal changes, as discussed later. It supports multi-user operation with role-based access control, so that permissions (like acknowledging or configuring alarms) can be restricted according to user roles [6] – a critical feature for large operations with hierarchical teams.

ARCHITECTURE OVERVIEW

IC@MS follows a **modular multi-tier architecture** focused on flexibility, scalability, and interoperability [7]. Architecture emphasizes separation of concerns: the frontend deals purely with user interaction, the backend provides a coherent API and orchestrates data flow, and the actual alarm detection logic is handled by the Tango device layer (PyAlarm/AlarmHandler), which can be seen as pluggable alarm evaluators. Such modularity not only improves maintainability but also enables portability – for example, if a facility uses a different alarm generator, IC@MS's backend could be extended with new interface, avoiding a complete project refactor. The design also makes it easy to introduce new capabilities (for instance, integrating a new notification channel or analytics tool) by adding microservices or extending APIs, rather than monolithic changes.

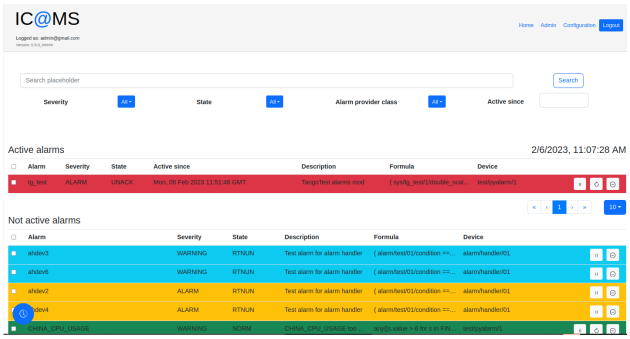

Figure 1: IC@MS home page.

Frontend

A client-side web application built with React.js that provides an intuitive, responsive interface for viewing alarms, acknowledging or shelving them, configuring alarm rules, and monitoring system status in real time (see Fig 1.). The web-based UI means operators and engineers can access the alarm system from any browser without specialized client software. It supports dynamic dashboards and forms for alarm configuration, making alarm management more user-friendly.

Backend

A server layer implemented in Python using the Flask framework, exposing a RESTful API (and GraphQL API) for all alarm management operations. This backend orchestrates communication between the frontend and the underlying control system. Crucially, it interfaces with Tango

* Work supported by S2Innovation.
† lukasz.zytniak@s2innovation.com

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-THPD058

alarm sources: it connects to PyAlarm and AlarmHandler Tango device servers to retrieve and update alarm definitions and statuses. The backend uses the PyTango library to subscribe to Tango events and device attributes, enabling real-time alarm state updates to be pushed to the frontend. Business logic in the backend handles alarm rule evaluation (delegated to the Tango devices), alarm state aggregation, user acknowledgments, and notification dispatching. The use of a REST/GraphQL API ensures that the system is not only modular but also accessible to external tools or scripts (for example, other applications can query alarm status or inject new alarm definitions through these APIs).

Database

A MongoDB database is used to store configuration and historical data. Alarm definitions (names, formulas, severities, etc.), user preferences, and alarm history (acknowledgment logs, occurrences) are recorded in MongoDB for persistence. Historical storage in MongoDB (a document-oriented NoSQL DB) offers flexibility in the data schema and the ability to handle large volumes of log entries (thousands to millions of alarm events) efficiently. Additionally, IC@MS leverages the existing Tango archiving system (HDB++ via the PyTangoArchiving API) to obtain historical values of device parameters related to alarms. This means that instead of duplicating the archiving functionality, IC@MS queries the Tango archival database for time-series data (e.g. to plot the trend leading up to an alarm). Only alarm-specific records (like when an alarm went on/off, who acknowledged it, etc.) are stored in MongoDB by IC@MS, while raw signal history remains in the facility's central archive.

Integration Layer

This layer encompasses the adapters and connectors to the underlying control system. In a Tango installation, the integration layer uses PyTango to communicate with any Tango device and subscribe to attribute events, and it specifically manages connections to the AlarmHandler and PyAlarm device servers which serve as alarm "engines". The integration layer is also where TangoGraphQL comes into play – TangoGraphQL is a GraphQL interface to Tango Controls that IC@MS employs for querying device metadata and states in a flexible manner (for example, the frontend can request alarm lists or device properties via GraphQL queries). These integration points are abstracted behind APIs. Thanks to this, the core IC@MS logic is decoupled from Tango specifics. This separation makes the system easier to adapt to other frameworks. (For instance, one could envision writing a different integration module to connect IC@MS to EPICS or another control system, while leaving the frontend and database unchanged.).

Deployment

IC@MS is delivered as a set of lightweight Docker containers, each encapsulating a part of the system (frontend, backend, database, etc.). Containerization makes it straightforward to deploy on various infrastructure – whether on-premises or in cloud environments – and to scale out the system if needed. For sites requiring high availability or distribution across multiple servers, IC@MS supports orchestration via Kubernetes (with provided Helm charts to simplify deployment). This allows the alarm service to be replicated, load-balanced, or placed close to local control networks as appropriate. In addition, the use of Docker/Kubernetes eases integration into existing CI/CD pipelines and infrastructure-as-code setups, so updates to the alarm system can be rolled out in a controlled, consistent manner.

FEATURES AND FUNCTIONALITY

IC@MS provides a comprehensive suite of features to cover the alarm management lifecycle from definition through notification and analysis.

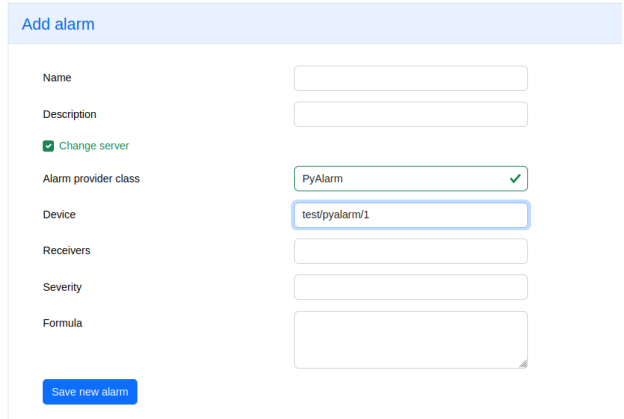

Figure 2: Add alarm form.

Alarm Definition & Configuration

Authorized users can define new alarms or modify existing ones through the IC@MS UI. An alarm is typically defined by a formula or condition involving one or more Tango device attributes (for example, a logical expression combining readings and states) – leveraging the underlying Tango Alarm Handler or PyAlarm to evaluate the condition continuously. The web interface allows entering these formulas or simple threshold rules, along with metadata like alarm description, severity level (priority), and associated tags or categories. Alarms can be added (see Fig 2.), updated, or removed on the fly via the GUI or programmatically via the API (see Fig 3. and Fig 4.).

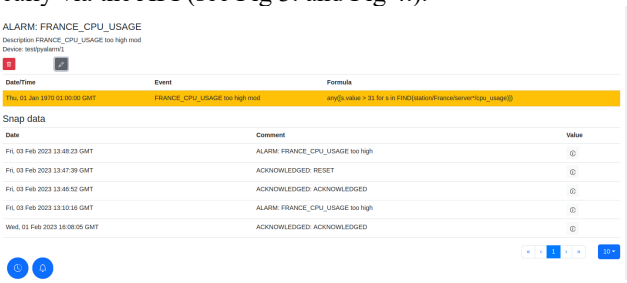

Figure 3: Alarm details page.

IC@MS supports grouping alarms by category, system, or user-defined tags, as well as defining hierarchical relationships. This grouping is useful in complex facilities to

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-THPD058

organize the alarm list (for instance, grouping all vacuum system alarms together or creating parent-child relations). All configuration changes persisted in the database and propagated to the Tango alarm devices in real time.

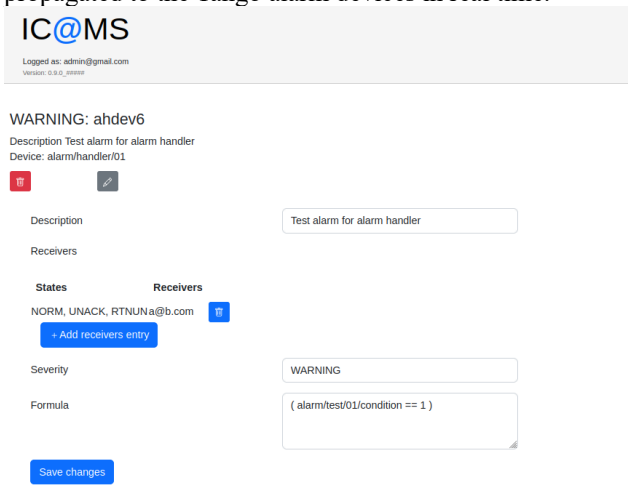

Figure 4: Alarm details page.

Real-Time Monitoring & Acknowledgment

A core feature of IC@MS is the live alarm dashboard. The frontend displays the status of all active alarms (and can also show recently cleared or shelved alarms), updating in real time as new alarms come in or existing ones change state. This is achieved by subscribing to Tango attribute events and alarm state changes via PyTango, so that any change in an instrument's status is reflected on the dashboard immediately. Operators can acknowledge alarms through the UI, which sends the acknowledgment to the backend and onward to the Tango alarm device (marking the alarm as acknowledged per the Tango alarm logic). The system supports standard alarm handling behaviors such as **latching** (alarm stays active until acknowledged even if the condition clears) and can enforce acknowledgment comments if required (to capture operator feedback on what was done).

IC@MS's GUI also allows filtering the view by alarm severity, system, or other criteria, and supports sorting and searching to help operators focus on the most critical alarms. In summary, the platform serves as a centralized alarm console where multiple users can concurrently monitor and act on alarms. Role-based access control ensures that, for example, only authorized roles can perform certain actions (like disabling an alarm or editing its parameters). This multi-user, multi-role design aligns with best practices in control room operations and the IEC 62682 guidelines for alarm management responsibility.

Notification & Automated Actions

IC@MS can be configured to deliver alarm notifications through various channels to ensure that the right personnel are informed of critical events. Built-in support exists for email (see Fig 5.) and SMS/text message notifications, which can be tied to specific alarm conditions or severity levels. For example, a high-severity alarm can be set to immediately email a list of on-call experts and send an SMS alert to a duty phone (see Fig 6.).

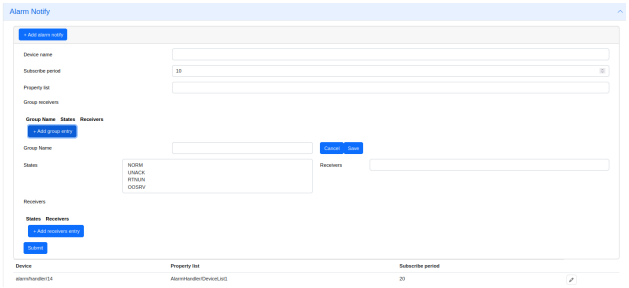

Figure 5: Alarm Notify.

The notification rules (who gets notified for what alarm and how) are configurable via the UI's notification settings module (often including features like scheduling – e.g. only page certain staff after hours, etc.). Under the hood, IC@MS triggers these notifications by hooking into the alarm state changes generated by PyAlarm/AlarmHandler – when an alarm enters the active state or clears, the backend can execute the associated notification actions. In addition to passive notifications, IC@MS (through PyAlarm) supports automatic **remedial actions** in response to alarms. The underlying PyAlarm device server is capable of launching external commands or calling Tango device commands when an alarm fires. IC@MS provides a user-friendly way to configure these automated actions (for instance, to perform an emergency shutdown if a critical threshold is exceeded, or to switch on a warning light). This capability means the alarm system is not just for monitoring but can also actively assist in mitigating issues. All actions (whether notifications or automated responses) are logged for traceability.

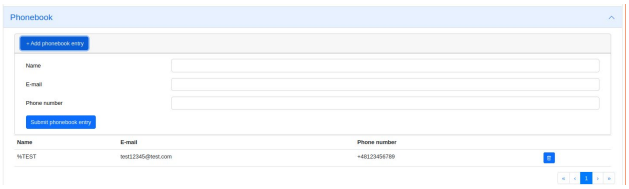

Figure 6: Phonebook.

Historical Alarm Data & Analysis

Beyond real-time monitoring, IC@MS offers tools to review and analyze alarm history, which is important for diagnosing recurring issues and improving the alarm configuration. Every alarm event (activation, acknowledgment, return-to-normal) is recorded with a timestamp and relevant details in the MongoDB alarm history collection. Users can browse this history via the interface (see Fig 7.), filtering by time range, alarm name, severity, etc., to find patterns (e.g. an alarm that frequently oscillates or a period with an alarm flood). More powerfully, IC@MS integrates with Tango's **HDB++ archival database** to retrieve historical trends of the process variables related to alarms. Through the GraphQL or REST API, the frontend can query archived data (for example, the temperature readings leading up to a temperature alarm) and display them as

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-THPD058

charts. This provides context for alarms – operators can see if a reading was steadily climbing or spiking at the time of the alarm, correlate multiple signals, or confirm if an alarm was spurious.

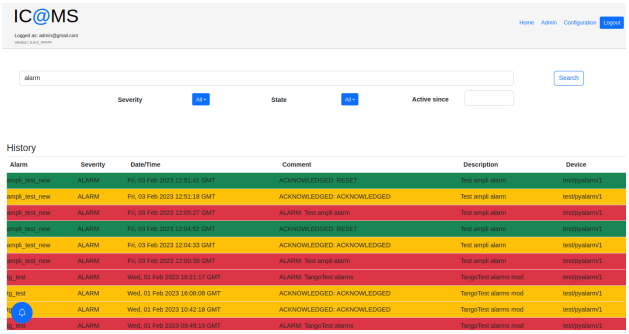

Figure 7: Alarms history page.

The system can also generate summary reports, such as the number of alarms per day or the top ten most frequent alarms over a month, helping engineers identify nuisance alarms or candidates for reconfiguration. By incorporating historical analysis, IC@MS supports the **alarm life-cycle management** process (as per IEC 62682) by supplying data for the rationalization and maintenance stages (e.g., deciding if an alarm threshold should be adjusted to reduce noise).

System Integration & Extensibility

IC@MS is designed to fit into the broader control system environment of a facility. It exposes a well-documented REST API (with interactive docs via Swagger, see Fig 8.) and a GraphQL endpoint, which allow other applications or scripts to interface with the alarm system. For example, a high-level physics application could query the alarm status before initiating a beam experiment, or an operations logging system could pull alarm occurrences to include in shift reports.

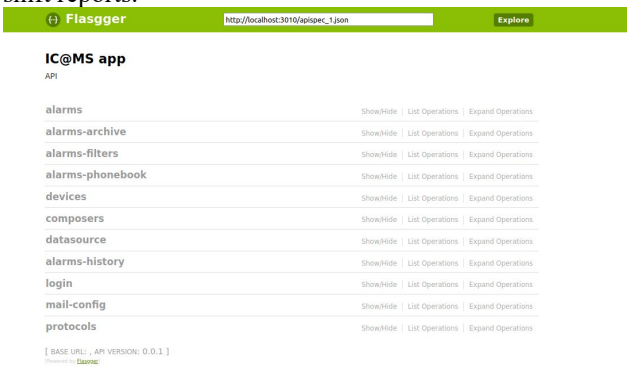

Figure 8: Swagger.

The APIs also make it possible to integrate IC@MS with non-Tango systems – for instance, an EPICS-based subsystem could send its alarms to IC@MS through the REST interface, allowing IC@MS to serve as a unified alarm portal. The modular architecture means new **alarm source adapters** or plugins can be developed as needed. While the current implementation is tightly integrated with Tango (using PyAlarm/AlarmHandler), the core concepts (alarm definitions, states, notifications) are not Tango-specific. Thanks to this modularity, IC@MS can be adjusted to other frameworks with relatively small effort – essentially by replacing or extending the integration layer. This flexibility ensures that the investment in IC@MS can be leveraged even if a facility's control system evolves beyond Tango. In addition, IC@MS's containerized deployment and configuration can be customized to different IT environments, and its open APIs future-proof it for integration with emerging technologies or services (see Fig 9.).

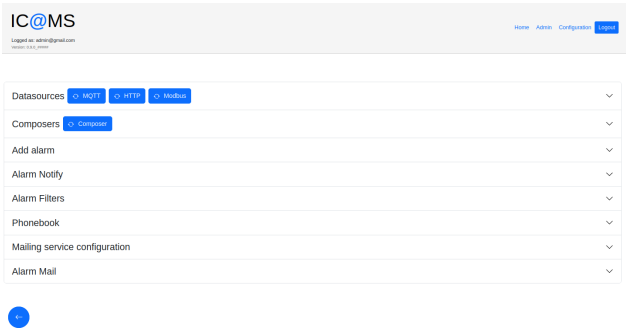

Figure 9: Configuration page.

DEPLOYMENT AND MODULARITY ADVANTAGES

One of the strengths of IC@MS is in how easily it can be deployed and adapted in different environments. The entire system is delivered via Docker images, which encapsulate all necessary dependencies. A facility can deploy IC@MS on a single server or in a distributed manner on a Kubernetes cluster. Scaling the system (for example, to handle a very large number of alarms or to isolate alarm management for different sub-facilities) is as simple as launching additional container instances and configuring them to listen to the relevant Tango devices. This cloud-native deployment model not only simplifies installation but also makes it feasible to offer IC@MS as a service (for instance, a centralized alarm service for multiple experimental stations, possibly running on virtualized infrastructure or in the cloud). Data persistence is handled by MongoDB volumes, which can be replicated or backed up using standard database tools.

The modular design also underpins IC@MS's adaptability. Each component (frontend, backend, integration, database) could be modified or replaced without overhauling the entire system, as long as the interfaces remain consistent. For example, if a facility prefers a SQL database, the MongoDB layer could be swapped out with a relational database with some changes to the data access module, leaving the rest of the system unaffected. Similarly, to integrate with a different control system, one could implement an alternate integration service that translates that system's alarms into IC@MS's expected format. The existing Tango-focused modules can serve as a template for such extensions. The use of standard protocols (REST, WebSockets, GraphQL) means IC@MS can communicate with virtually any modern control middleware or enterprise system. In short, the technology-agnostic approach in the

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-THPD058

higher layers of IC@MS ensures that it is not locked into Tango – Tango is currently the primary use-case due to its prevalence in the facilities where IC@MS was first deployed, but the concept and software structure are general. This positions IC@MS as not just a Tango tool, but a potential framework-agnostic alarm management platform with minimal adaptation.

Another practical advantage of IC@MS's design is maintainability and community support. By leveraging open-source components (MongoDB, Flask, React) and aligning with the Tango community's tools, the system benefits from ongoing updates in those ecosystems. Users of IC@MS do not have to maintain a proprietary GUI or custom protocol; instead, they have a solution built on widely used software with a large support base. The Tango integration means that all of Tango's features (like its device directory, event system, and security mechanisms) are immediately available to IC@MS. Sites that already use Tango will find IC@MS slots in naturally, and those using other systems can still adopt IC@MS by deploying Tango Alarm Handlers or PyAlarms as bridges (since those Tango alarm servers can run independently for the purpose of alarm evaluation).

CONCLUSION

IC@MS represents a modern approach to alarm management in scientific control systems, combining the reliability of existing Tango alarm mechanisms with the benefits of contemporary web software. It provides a unified, vendor-neutral interface for operators to supervise alarms coming from thousands of devices in real time, with capabilities for historical analysis and automated response that improve situational awareness and operational efficiency. The platform has already been successfully deployed in environments like particle accelerators and synchrotron light sources, where it has managed on the order of thousands of concurrent alarms without performance issues. This production experience has demonstrated the system's scalability and robustness in mission-critical scenarios.

Thanks to its modular architecture and adherence to industry standards, IC@MS is highly adaptable. Even though it was initially built for Tango Controls, it can be readily customized for use in any scientific or industrial facility – potentially integrating with alternative control frameworks – with minimal changes to core components. This flexibility, combined with containerized deployment, means that IC@MS can be tailored to different sites' needs (from small lab setups to large, distributed organizations) and can evolve with those needs. In summary, IC@MS provides a comprehensive alarm management solution that unifies disparate alarm sources under one roof, simplifies the operator's experience, and complies with modern best practices in alarm handling. Its design ensures that any facility can adopt it as a turn-key alarm console, or extend it to fit their ecosystem, thereby improving their overall control system reliability and responsiveness.

REFERENCES

[1] Tango Controls – An object-oriented control system framework for scientific instruments, `http://www.tango-controls.org`

[2] L. Pivetta, "Development of the Tango Alarm System," in *Proc. ICALEPCS 2005*, Geneva, Switzerland, Oct. 2005, paper WE3B.1-70.

[3] S. Rubio-Manrique *et al.*, "Extending Alarm Handling in Tango," in *Proc. ICALEPCS 2011*, Grenoble, France, Oct. 2011, pp. 63–65.

[4] S. Rubio-Manrique, F. Becheri, G. Cuni, D. Fernandez-Carreiras, C. Pascual-Izarra, and Z. Reszela, "PANIC, a Suite for Visualization, Logging and Notification of Incidents", in *Proc. PCaPAC'14*, Karlsruhe, Germany, Oct. 2014, paper FCO206, pp. 243-245.

[5] S. Rubio-Manrique, G. Cuni, D. Fernandez-Carreiras, and G. Scalamera, "PANIC and the Evolution of Tango Alarm Handlers", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 170-175. `doi:10.18429/JACoW-ICALEPCS2017-TUBPL03`

[6] Ł. Żytniak *et al.*, "IC@MS - modular and containerized web-based alarm management system", in *Proc. IPAC'23*, Venice, Italy, May 2023, pp. 4269-4270. `doi:10.18429/JACoW-IPAC2023-THPA129`

[7] Ł. Żytniak, M. Gandor, P.P. Goryl, J.T. Kowalczyk, M. Nabywaniec, S. Rubio-Manrique, "IC@MS - Web-Based Alarm Management System," in *Proc. PCaPAC2022*, Dolní Brežany, Czech Republic, Oct. 2022, pp. 48-50. `doi:10.18429/JACoW-PCaPAC2022-THPP8`