ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEPD043

PLC INTEGRATOR: A MODERN TOOL FOR PLC-EPICS INTEGRATION AT ESS

A. Fontoura†, S. Jäger, A. Rizzo, H. Ligander, J. Choi, E. Foy, ESS ERIC, Lund, Sweden

Abstract

The *PLC Integrator* is a new tool developed at the European Spallation Source (ESS) to modernize the integration of PLC-based control systems with the EPICS framework. It replaces the legacy *PLC Factory*, removing outdated dependencies such as the CCDB database, while introducing support for modern technologies like OPC-UA and Beckhoff ADS. The software retains key features familiar to integrators - such as automatic PLC-IOC communication code generation, alarm handling, and archiving - but is built on a sustainable, extensible, and easier-to-maintain foundation. By adopting JSON files instead of the older *.def* format, it improves standardization, enables automatic validation, and increases memory efficiency on PLCs. This article presents the context and motivations behind its development, describes the architecture and main features of the PLC Integrator, highlights the challenges faced during implementation, and discusses its impact at ESS as well as the path forward.

INTRODUCTION

The European Spallation Source (ESS), under construction in Lund, Sweden, is one of the world's largest scientific infrastructure projects, aiming to become a global leader in neutron scattering research. To achieve this, its operation relies on highly robust, scalable, and interoperable automation systems capable of handling the complexity of large-scale scientific instruments. Within ESS, the ICSHWI group (Instrumentation & Control Systems Hardware and Integration), and specifically its Automation Section, plays a central role by bridging Programmable Logic Controllers (PLCs) with the EPICS (Experimental Physics and Industrial Control System) framework [1]. EPICS underpins communication and supervision across thousands of devices. For years, this integration relied on an in-house tool called PLC Factory [2], which simplified tasks such as generating PLC-IOC communication [3] code for Siemens controllers, managing alarms, and supporting device hierarchies. While effective early on, the tool accumulated significant technical debt over time. Its main limitations were:

- Reliance on an outdated version of Python [4].
- Dependence on the CCDB database [5], which was scheduled for decommissioning.

Faced with these limitations, ESS needed a new solution that would ensure continuity, remove obsolete dependencies, and prepare the infrastructure for future challenges. This need gave rise to the PLC Integrator.

† adalberto.fontoura@ess.eu

DEVELOPMENT AND IMPLEMENTATION

The *PLC Integrator* was designed to be backward-compatible with existing workflows, while offering completely modernized architecture (Fig. 1).

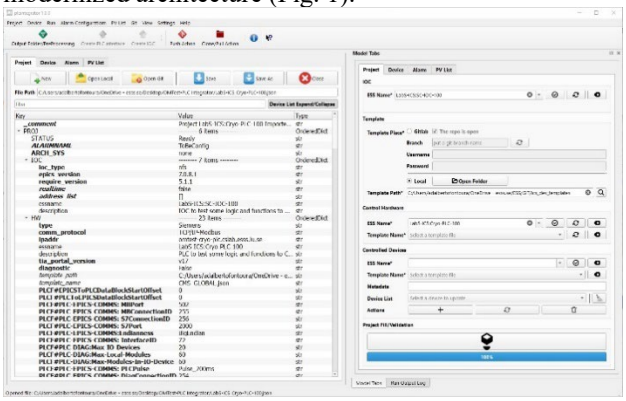

Figure 1: PLC Integrator UI.

Interface and Platform

Its graphical interface was built from the ground up using PySide6 [6], resulting in a modern, responsive, and intuitive application. The goal was to shorten the learning curve for engineers familiar with PLC Factory, while also adding new features and improved usability.

Modular Architecture

All functions were restructured into independent modules (Fig. 2), each responsible for a specific task such as template management, preprocessing, or code generation.

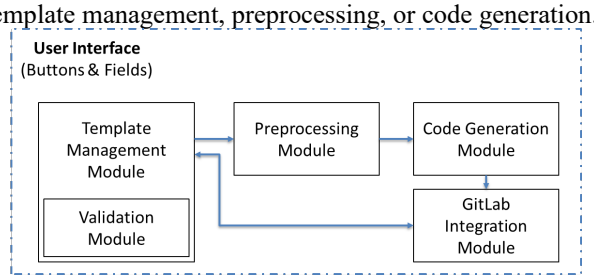

Figure 2: Modular Architecture

- **Template Management Module:** Manage Device Templates and Project Templates.
- **Validation:** Performs real-time validation during configuration, checking user inputs, JSON syntax, and PV consistency, running continuously in the background rather than only at preprocessing.
- **Preprocessing Module:** Validate data before code generation. Detect errors in PV names (missing, duplicated, or invalid).

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEPD043

- **Code Generator Module:** Automate the creation of PLC programs and IOC modules.
- **GitLab Integrator Module:** Connect project workflows to the ESS version control system.
- **User Interface (UI):** Although not a backend module, the graphical interface is the central hub of the PLC Integrator: Connects all modules via action buttons (e.g., Generate Code, Validate Project, Push to GitLab).

PLC Integrator is structured as a collection of independent modules. Each module handles a specific part of the workflow, while the user interface orchestrates them seamlessly, making the tool powerful, easier to maintain, more scalable, and clearer to develop further yet easy to use.

Continuity and Compatibility

Despite its modernization, the PLC Integrator preserves the core logic (Fig. 3) of its predecessor [2]. This ensures smooth migration for integrators, allows reuse of established workflows, and maintains compatibility with previously generated data.

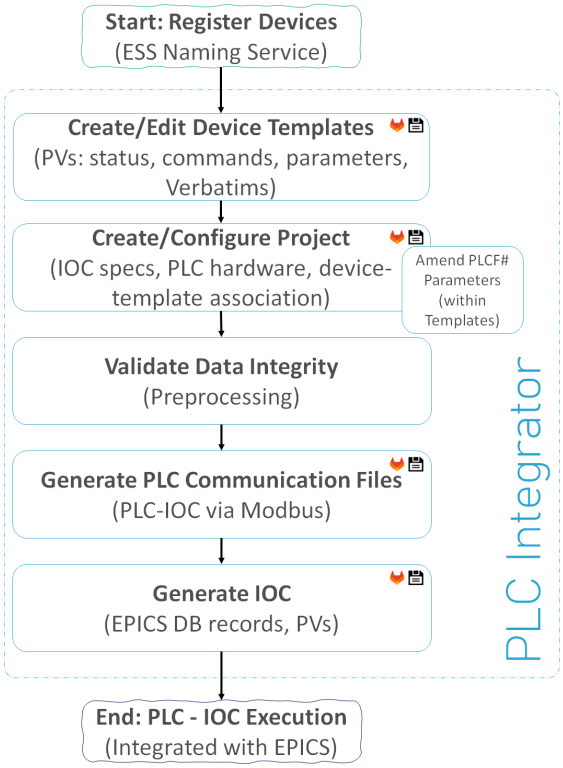

Figure 3: Core logic.

From .def to .json

A significantly important part of the PLC Factory platform are definition files, they are configuration files used in legacy systems to describe devices, process variables (PVs), and control parameters. They served as a reference for generating PLC-IOC code in ESS, but they had limitat ions such as lack of standardization, difficulty in automatic validation, and inefficient memory usage, which led to their replacement by JSON files in PLC Integrator. Replacing the legacy *.def* format with **JSON** was a game-changer. JSON files provide a rigid, well-defined structure, enabling automatic validation, syntax error detection, and clearer organization.

This shift also simplifies future growth: new fields or functions can be added naturally and consistently without breaking workflows.

FEATURES

The PLC Integrator was designed to simplify the work of engineers at ESS [7], making PLC–EPICS integration consistent, efficient, and easy to manage. It acts as a central platform for configuring, validating, and generating integration code between PLCs and EPICS, reducing much of the repetitive manual work that was once unavoidable. The general workflow (3) follows these steps:

- **Device Registration:** Devices must first be registered in the ESS Naming Service [8], ensuring each has a unique and consistent identifier.
- **Device Template Creation:** For each device, a Device Template is created defining PVs (Process Variables), commands, parameters, and optional *PLCF#* tags or verbatims***. Templates can be reused across projects and shared (i.e. *via GitLab*).
- **Project Creation:** Configure IOC details, PLC/hardware information, and link devices to their templates. Edit PLCF# parameters, run data preprocessing to validate consistency, and generate files for PLC–IOC communication (initially via Modbus). Automatically create IOC files, including .db databases with PVs.
- **Project Saving:** Completed projects can be stored locally for future updates and documentation.
- **GitLab Integration:** Entire projects can be pushed to GitLab repositories, ensuring version control and team collaboration.
- **Execution:** With files generated, integrators can: Run the IOC and establish communication with EPICS. Deploy PLC communication blocks, enabling efficient and standardized PLC–IOC interaction.

This workflow shows how the PLC Integrator turns complex, repetitive integration tasks into a logical, assisted, and standardized process, tightly linked with essential ESS systems like the Naming Service and GitLab.

Device Templates

Device Templates (Fig. 4) are the foundation of integration. Each template defines process variables (PVs), commands, and parameters associated with a device. These can be categorized as Status, Command, or Parameter, and can include specific tags *PLCF#* and *verbatims* for fine adjustments. By reusing templates across multiple projects, engineers achieve greater consistency while minimizing errors. Templates can also be stored locally or shared via GitLab repositories, promoting collaboration across teams.

* PLCF is a simple embedded domain-specific language for use in template files to substitute parameters value (e.g. EGU = K).
** Verbatim are literal texts containing EPICS records that will be added to the IOC EPICS db and PLC code.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEPD043


```
{
    "_comment": "",
    "DEV": {
        "GlobalVariables": "",
        "ExternalValidity": "",
        "Status": {
        "analog": [
        "digital": [
        "time": [
        "enum": [
        "bitmask": [
        "string": [
        },
        "Command": {
        "analog": [
        "digital": [
        "time": [
        "enum": [
        "bitmask": [
        "string": [
        },
        "Parameter": {
        "analog": [
        "digital": [
        "time": [
        "enum": [
        "bitmask": [
        "string": [
        },
        "PLCF": {
        "Verbatim": [
    }
}
```


Figure 4: Device Template file (.json format).

Project Templates

Project Templates (Fig. 5) bring together multiple Device Templates into a coherent project configuration. They ensure that each project is clearly documented, with devices, parameters, and connections organized systematically. Compatibility with the ESS Naming Service [8] guarantees unique and consistent identifiers for each device, which is vital for large-scale integration.


```
{
    "_comment": "",
    "PROJ": {
        "STATUS": "Draft",
        "ALARMNAME": "",
        "ARCH_SYS":  "none",
        "IOC": {
            "ioc_type": "nfs",
            "epics_version": "7.0.8.1",
            "require_version": "5.1.1",
            "realtime": false,
            "address_list": "[]",
            "essname": "",
            "description": ""
        },
        "HW": {
            "type": "",
            "comm_protocol": "",
            "ipaddr": "",
            "essname": "",
            "description": "",
            "template_path": null,
            "template_name": null
        },
        "CTRL": {
            "TEMPLATEPLACE": "",
            "TEMPLATEPATH": "",
            "DEVICE": [
            ]
        }
    }
}
```


Figure 5: Project Template file (.json format).

Preprocessing and Automated Code Generation

One of the biggest innovations in the PLC Integrator is the preprocessing mechanism (Fig. 6) [9]. Before any final files are created, the system checks for consistency and flags potential errors, such as invalid PV names, missing parameters, or incorrect hardware addresses. Only after this validation does the software automatically generate:

- Optimized PLC programs, designed for efficient memory usage.
- IOC files, including databases with all PVs needed for EPICS integration.

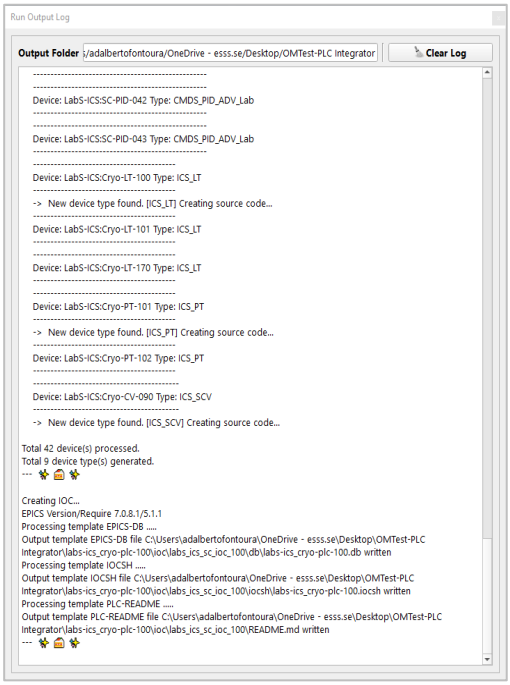

Figure 6: Processing and code generation log.

Real-Time Data Validation

Validation is not limited to preprocessing - it runs continuously during project setup. This early detection prevents many errors from reaching the testing phase, saving time and effort. It also ensures that all project data remains in sync with the Naming Service, avoiding conflicts or duplications.

GitLab Integration

With native GitLab integration, both projects and templates can be version-controlled automatically. This provides full traceability, supports collaborative work, and strengthens long-term maintainability.

CHALLENGES OF IMPLEMENTATION

Developing the PLC Integrator was not just about writing new software, it meant **reshaping the entire integration ecosystem** at ESS Automation Section. This transition brought technical, operational, and organizational challenges.

Designing a Modern Interface

Building the interface from scratch required balancing ease of use with advanced functionality [9].

- It had to feel familiar enough for engineers coming from PLC Factory, while also supporting new workflows.
- External tools such as GitLab and the Naming Service had to be integrated seamlessly into the UI, so tasks

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEPD043

like pushing projects or syncing devices were intuitive.
- Usability testing carried many iterations, extending the development timeline but resulting in a more robust and user-friendly tool.

Migrating Projects from CCDB

Decommissioning the CCDB database was one of the toughest challenges. CCDB projects relied on specific formats and REST APIs, which the new tool could not support directly. To ensure continuity:
- Conversion scripts were developed to translate legacy data into JSON.
- Devices and hierarchies were restructured to fit the new modular model.
- All migrated projects were validated to confirm synchronization with the Naming Service and compatibility with IOC generation.

This process required close collaboration between developers and users, as any data loss could have disrupted critical operations.

Transitioning from .def to .json

Moving from *.def* to JSON was both necessary and challenging. Legacy files often contained:
- Inconsistent or redundant structures.
- Poorly formatted configurations that were hard to translate automatically.
- Inefficient memory usage that strained PLC performance.

The solution included:
- Automated conversion tools with built-in validation.
- Strict normalization rules to enforce consistency.
- A redesigned code generation logic to improve memory efficiency.

User Adoption

Beyond technical hurdles, adoption by engineers posed its own challenges. Users accustomed to PLC Factory had to learn new concepts such as JSON templates and workflows. Training sessions, clear documentation, and hands-on support during rollout were essential to smooth the transition.

IMPACT AND FUTURE

The *PLC Integrator* has already delivered clear benefits to ESS:
- Reliability and standardization: reduced risk of configuration errors and improved consistency across projects.
- Efficiency: optimized memory usage on PLCs and faster code generation.
- Reduced technical debt: removal of outdated dependencies like CCDB and legacy Python.
- Long-term sustainability: modular architecture ensures the tool can evolve with future needs.

Looking ahead, the planned integration of OPC UA [10] and Beckhoff ADS [11] will be crucial. OPC UA brings cross-platform interoperability and strong security, while ADS offers low-latency, direct integration with Beckhoff hardware. Together, they will further position ESS as a leader in modern scientific automation, with scalable, secure, and future-ready workflows.

CONCLUSION

The *PLC Integrator* is more than just a replacement for PLC Factory—it represents a generational leap in PLC-EPICS integration at ESS. Its modular design, modern interface, and support for advanced protocols make it a sustainable, forward-looking solution.

By cutting technical debt and improving operational efficiency, ESS ensures its automation infrastructure is not only ready for the start of operations, but also for decades of groundbreaking scientific research to come.

REFERENCES

[1] A. Rizzo et al., "Full Stack PLC to EPICS Integration at ESS", *in Proc. ICALEPCS'23*, Cape Town, South Africa, Oct. 2023, pp. 1216-1220. doi:10.18429/JACoW-ICALEPCS2023-THMBCM011

[2] G. Ulm, F. Bellorini, D. P. Brodrick, R. N. Fernandes, N. Levchenko, and D. P. Piso, "PLC Factory: Automating Routine Tasks in Large-Scale PLC Software Development", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 495-500. doi:10.18429/JACoW-ICALEPCS2017-TUPHA046

[3] S. Pande *et al.*, "CODAC Standardization of PLC Communication", in *Proc. ICALEPCS'13*, San Francisco, CA, USA, Oct. 2013, pp. 1097-1099.

[4] Python webpage, https://www.python.org/

[5] T. Korhonen, "EPICS IOC Configuration using the CCDB Toolchain", ESS, Lund, Sweden, Rep. ESS-2756125, Nov. 2020. https://chess.esss.lu.se/enovia/tvc-action/validVersion?versionObjectId=21308.51166.36096.3388&inline=false

[6] Pyside6 webpage, https://pypi.org/project/PySide6/

[7] D. Piso *et al.*, "ESS PLC Controls Strategy", in *Proc. IPAC'15*, Richmond, VA, USA, May 2015, pp. 3066-3068. doi:10.18429/JACoW-IPAC2015-WEPMN061

[8] K. Rathsman, G. Trahern, J. Malovrh Rebec, M. Rescic, and M. Vitorovic, "ESS Naming Convention", in *Proc. IPAC'13*, Shanghai, China, May 2013, pp. 3222-3224.

[9] M. D. Dutour, "Software Factory Techniques Applied to Process Control at CERN", in *Proc. ICALEPCS'07*, Oak Ridge, TN, USA, Oct. 2007, paper TPPA03, pp. 87-89.

[10] OPC UA webpage, https://opcfoundation.org/about/opc-technologies/opc-ua/

[11] Beckhoff ADS webpage, https://infosys.beckhoff.com/english.php?content=../content/1033/tcinfosys3/11291871243.html&id=