IFMIF-DONES CONTROL SYSTEMS AND TESTING PLATFORM

A. Naranjo Jiménez*, C. Carvajal Almendros, M. J. Gutiérrez,
R. Lorenzo Ortega, L. Maindive, I. Podadera, M. Weber
IFMIF-DONES España, Escúzar, Granada, Spain

Abstract

The IFMIF-DONES neutron source, under construction in Granada, Spain, has high availability targets for its systems. In order to evaluate the architectural and technological choices being made for the control systems, a dedicated testing platform is being developed. Key aspects under evaluation include the EPICS IOC deployment model (container-based) and orchestration solution (currently evaluating Proxmox HA, Docker Swarm and Kubernetes). Integration of existing DONES prototypes into the testing platform is also envisaged. The platform delivers value throughout all the facility's lifecycle. This contribution will report on the design and current progress of the testing platform.

INTRODUCTION

IFMIF-DONES (International Fusion Materials Irradiation Facility - Demo Oriented Neutron Source) [1] is a critical pillar in the global fusion roadmap, designed to test and qualify materials under extreme neutron irradiation. To ensure safe and continous operation of the facility, the Control Systems play a vital role [2]. While manual deployment is beneficial in the early stages, allowing fast development cycles, these methods are inherently prone to human error, lack version control, and often lead to inconsistencies between development, testing, and production environments.

To address these challenges, it was decided to develop CICStest, a Central Instrumentation Control Systems (CICS) [1, 2] test platform. This platform adopts "Control as Code" principles, integrating industrial control standards with modern DevOps practices. Using Infrastructure as Code (IaC) through Ansible, containerization via Docker, and automated orchestration with GitLab CI/CD, CICStest provides a high-fidelity, reproducible environment for validating architectural choices, evaluating different technologies, developing software, human factors evaluation and training.

This paper is organized as follows: First, it presents the hardware infrastructure for CICStest; then, it details the software configuration based on the Phoebus ecosystem; later, it describes on the automated deployment workflow; after that, it presents a Software Development Kit (SDK) developed using the availability of CICStest; and finally, it presents the conclusions and future development lines for CICStest.

HARDWARE CONFIGURATION

The CICStest platform is hosted on a dedicated infrastructure designed to simulate a controlled industrial environment. The core of the system consists of a dedicated Dell PowerEdge R750XS server, which provides the necessary computational resources to achieve virtualization and container orchestration. This setup allows for virtualization and validation of the control environment while the final IFMIF-DONES Control System and networking infrastructure is being developed and constructed. By decoupling the control system development from the physical site availability, the platform ensures that architectural standards and network configurations are pre-defined and tested in advance.

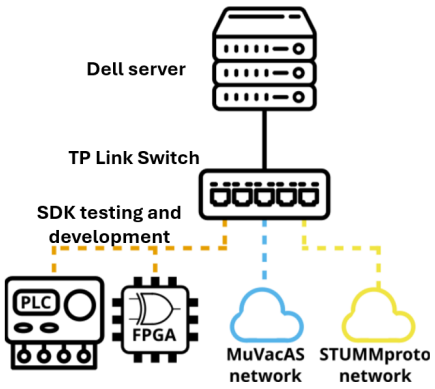

Figure 1: Visual scheme of the hardware configuration of the CICStest server, aiming a future connection to different hardware stations and prototypes.

To manage local traffic and field device communication, the setup includes a TP-Link TL-SG2428P switch, which ensures high-speed data transfer and VLAN segmentation if required. The platform is integrated into the broader infrastructure via a dedicated uplink to the University of Granada (UGR) network, allowing for remote management of different Local Instrumentation and Control Systems (LICS) [2], as shown in Fig. 1.

SOFTWARE CONFIGURATION

The CICStest software ecosystem has a modular, containerized architecture based on the EPICS (Experimental Physics and Industrial Control System) framework. The architecture is divided into three functional layers: Client, Services, and Standard Services [3], represented as three different columns in the mentioned order in Fig. 2.

Client and Control Services

The top layer features Phoebus as the primary integrated control system cockpit, providing a human-machine interface for control, monitoring, and alarm visualization. Supporting this, a suite of dedicated services is deployed:

* naranjo1911@gmail.com

- Alarm Management: Comprising the Phoebus Alarm Server and Alarm Logger to handle and record system faults.
- Data Archiving: The EPICS Archive Appliance is utilized for long-term storage of process variables.
- Operational Logs and Configuration: Save & Restore for managing machine snapshots; and integration of Phoebus OLog and MongoDB for managing and storing electronic logging, which is still in development.

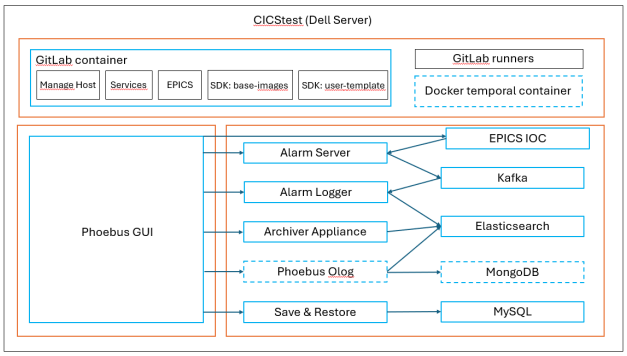

Figure 2: Logical architecture of the CICStest software stack, adapted from [3]. The orange squares represents the Proxmox Virtual Machines (VMs), while the blue squares represent the Docker containers.

Middleware and Data Persistance

The backbone of the system relies on a modernized messaging and database layer:

- Message Broker: from Phoebus 4.7.2, Apache Kafka 4 is used, eliminating the Zookeeper dependency.
- Persistence Layer: Data is managed across multiple specialized databases, including Elasticsearch for logs and alarm indexing, and MySQL for relational configuration data.

INFRASTRUCTURE DEPLOYMENT AND CI/CD WORKFLOW

The deployment of the CICStest platform follows a "Control as Code" philosophy, leveraging modern DevOps tools to ensure reproducibility, scalability, and rapid recovery. Other members of the EPICS community are leveraging this philosophy as well [4–6].

Virtualization and Containerization Layer

The hardware resources are managed using Proxmox Virtual Environment, an open-source server management platform. Proxmox allows for the logical isolation of services into specialized VMs. In this architecture, VMs act as functional nodes that emulate the distributed nature of the final IFMIF-DONES control network. This approach allows for the simulation of different network segments and the allocation of dedicated resources (CPU, RAM, and I/O) to specific infrastructure roles, such as the GitLab orchestrator or the core EPICS server.

Furthermore, deploying services within VMs facilitates the execution of specialized tasks that require direct host interaction. A key example are the GitLab runners, which are installed at the VM host level rather than inside a container. This ensures the necessary permissions and performance to interact with the Proxmox API and the Docker daemon during complex deployment cycles.

In addition, Docker is used for service encapsulation. This hybrid approach of containers within VMs is justified by several factors:

- Consistency: Ensures the same environment across development and production.
- Isolation: Each EPICS service (IOCs (Input-Output Controllers), Archiver, Phoebus services) runs in an isolated user space, preventing dependency conflicts.
- Resource Efficiency: Containers share the host OS kernel, allowing for a higher density of services compared to traditional VMs.

Centralized Management: The GitLab and Ansible Hub

The deployment process is managed through a centralized GitLab instance running in a dedicated container. To ensure high performance and isolation during the build and deployment phases, GitLab Runners are installed directly on the host VM (outside the GitLab container), allowing them to execute heavy tasks and interact with the virtualization layer efficiently.

The workflow for provisioning and configuring the control infrastructure is as follows:

- Virtual Machine Templating: To minimize deployment time and ensure environment consistency, a "Gold Master" VM template is maintained within Proxmox. This template contains a pre-configured, minimal Debian environment ready for software provisioning.
- Automated Provisioning with Ansible: The deployment pipeline triggers Ansible playbooks that utilize the Proxmox API to clone the master template into new functional nodes. This approach guarantees that every testing instance starts from a known, clean state.
- Containerized Service Deployment: Once the VMs are provisioned, Ansible handles the installation of the Docker engine and pulls the required images (EPICS IOCs, Phoebus services, Kafka, etc.) from the GitLab Container Registry.
- CI/CD Pipeline Integration: Every modification to the control system configuration, from a new EPICS record to a database schema change, is managed via Git. A git push command automatically triggers the pipeline, which validates the code and updates the CICStest environment in real-time.

SOFTWARE DEVELOPMENT KIT FOR LICS STANDARDIZATION

Since IFMIF-DONES consists of multiple subsystems developed by different in-kind contributors, there is a significant risk of technical fragmentation in the control logic. This distributed development model poses a significant challenge for the integration and maintenance of the facility. To mitigate the risks of software fragmentation, it is essential to provide a unified framework that ensures that all the LICS are consistent with the overall control architecture of the project.

To address this need, the CICStest project has facilitated the building of a dedicated Software Development Kit, which contains a pre-configured EPICS IOC container with the necessary OPC UA libraries already set for its proper connection with any external LICS [7]. This SDK serves as a standardized template for external contributors, offering a modular and containerized environment based on Debian Linux. By providing this "ready-to-use" stack, the project ensures that any LICS developed, regardless of its origin, is pre-configured for agile EPICS deployment and features modern industrial connectivity under the OPC UA standard. This approach not only streamlines the integration phase but also that the final control infrastructure remains harmonious and maintainable throughout its operational lifetime.

The SDK was tested by deploying a virtual PLC and an EPICS IOC built from the provided template. Its connectivity to the different Phoebus services of CICStest was verified, ensuring correct integration within the control stack, reliable data exchange across the middleware layer, and proper interaction with supervision, archiving, and alarm management services in a representative operational environment.

CONCLUSION

The development of the CICStest platform has demonstrated the feasibility of applying the "Control as Code" philosophy in the control environment for IFMIF-DONES. The core achievement of this work lies in the orchestration of a complex control stack through the intensive use of Ansible in a continuous integration environment.

By defining the entire control infrastructure—from the operating system parameters to the deployment of individual EPICS IOCs—within Ansible playbooks, we eliminated the risks associated with manual configuration. This approach ensures that any instance of the testing platform is identical and easily reproducible. The synergy between Ansible and Proxmox allows for the automated cloning of predefined templates, providing a clean and standardized state for every test cycle, which is a critical requirement for validating transversal systems in large-scale facilities.

The implementation of GitLab CI/CD pipelines acts as the final orchestrator of this logic. Each time a developer modifies a control configuration or a deployment rule, the pipeline triggers the corresponding Ansible execution. This workflow does more than just deploy software; it validates the entire deployment logic in real-time. This automation significantly reduces the time between development and verification, allowing for rapid iterations of the Phoebus and EPICS services without compromising system integrity. In the future operational phase, this workflow will transition to automated deployment, ensuring that updates are first validated within CICStest as part of the CI process before being manually triggered during scheduled maintenance windows.

FUTURE WORK

The CICStest platform is designed for continuous evolution. Future developments will focus on three main strategic areas to reach full operational readiness:

- Completion of the Control Stack: Priority will be given to the full integration and validation of the remaining Phoebus services. This includes the deployment of the Alarm Logger for persistent fault auditing, the Channel Finder for dynamic directory services, and the inclusion of Phoebus Olog and MongoDB to support advanced metadata storage.

- Identity and Access Management (IAM): To meet the security and traceability requirements of a radiological facility, a centralized authentication and authorization system will be implemented. We plan to evaluate the integration of Authentik or Keycloak via OIDC/SAML protocols. Phoebus 6 will include native support for OAuth2 for all services, so this system will be tested as well [8]. This will allow for granular user management and secure access control across all platform interfaces, from the GitLab orchestrator to the Phoebus dashboards.

- Real environment integration: For a transition to hardware testing, to independent stations will be connected to CICStest, working as LICS, starting with a PLC, to validate standard industrial protocols. Subsequently, an FPGA will be integrated to test high-speed data acquisition and low-latency control loops within the automated deployment workflow.

In the longer term, the platform aims to become the central testing CICS environment, using the different prototypes in IFMIF-DONES as LICS. This involves the full incorporation of the LICS from MuVacAS (Multipurpose Vacuum Accident Scenarios) [9], STUMM-PROTO (Start-Up Monitoring Module Prototype) [10], and other future prototypes. Integrating these three systems into CICStest will be a major milestone toward a fully unified and standardized central control system.

REFERENCES

[1] A. Ibarra *et al.*, "The IFMIF-DONES project: preliminary engineering design", *Nucl. Fusion*, vol. 58, p. 105002, Aug. 2018. doi:10.1088/1741-4326/aad91f

[2] M. Capelli *et al.*, "The IFMIF-DONES control architecture: the state-of-the-art design of central and local control systems and communication networks", *Nucl. Fusion*, vol. 65, p. 122003, Sep. 2025. doi:10.1088/1741-4326/adc1dd

[3] Control System Studio, "Services Architecture", CS-Studio documentation. https://control-system-studio.readthedocs.io/en/latest/services_architecture.html

[4] K. Shroff and R. Lange, "Shared ansible roles for EPICS, Phoebus and tools", presented at Spring EPICS collaboration meeting 2026, Gif-sur-Yvette, France, Apr. 2026. https://indico.in2p3.fr/event/37441/contributions/172015/

[5] M. Giacchini, "RAPIDS: A ready-to-use EPICS stack for facilities without a controls team", presented at Spring EPICS collaboration meeting 2026, Gif-sur-Yvette, France, Apr. 2026. https://indico.in2p3.fr/event/37441/contributions/172309/

[6] R. Nicole, "Deploying every Phoebus service on a single machine with EPNix", presented at Spring EPICS collaboration meeting 2026, Gif-sur-Yvette, France, Apr. 2026. https://indico.in2p3.fr/event/37441/contributions/172289/

[7] R. Lange *et al.*, "Integrating OPC UA devices in EPICS", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 184-187. doi:10.18429/JACoW-ICALEPCS2021-MOPV026

[8] G. Weiss, "Phoebus eco system update", presented at Spring EPICS collaboration meeting 2026, Gif-sur-Yvette, France, Apr. 2026. https://indico.in2p3.fr/event/37441/contributions/172310/

[9] A. Sabogal *et al.*, "MuVacAS: Experimental setup for testing mitigation strategies against Loss of Vacuum Accidents in the IFMIF-DONES accelerator", *Fusion Eng. Des.*, vol. 222, p. 115473, Jan. 2026. doi:10.1016/j.fusengdes.2025.115473

[10] S. Becerril-Jarque *et al.*, "A real-sized start-up monitoring module prototype for comprehensive test and irradiation campaigns of miniaturized neutron detectors according to the IFMIF-DONES baseline", *Nucl. Mater. Energy*, vol. 40, p. 101712, Sep. 2024. doi:10.1016/j.nme.2024.101712