ISBN: 978-3-95450-248-6 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2025-THPS122

FGC TEST MANAGER: A FRAMEWORK FOR EXECUTING AND MONITORING SOFTWARE TESTS

F. von Albedyll*, P. Plutecki, R. Murillo Garcia, D. Arominski, CERN, Geneva, Switzerland

Abstract

Function Generator Controllers (FGCs) are key devices used in CERN's converter control systems to regulate and monitor the power converters that supply current to the magnets in the accelerator complex. The FGC Test Manager has been developed to ensure the reliability and enhance the quality assurance of the software that controls these devices. It encompasses the Python library `pyfgc_test_framework`, which provides an interface for test scripts to communicate with the FGC devices, and a web tool, which allows users to run tests on schedule and on-demand, assign tests to resources, review test results, and directly access logs. The web tool uses Vue 3 for the front end and FastAPI with a PostgreSQL database for the back end. Test execution is handled by the GitLab Pipeline API, which executes pipelines directly in the repository containing the tests. This paper presents the design and functionality of the FGC Test Manager and the improvements it brings to the quality assurance of CERN's converter control systems.

BACKGROUND

Ensuring the reliability and functionality of software controlling critical devices in CERN's accelerator complex is essential to maintaining operational efficiency and safety, as well as ensuring availability and quality assurance. The converter controls software team plays a central role in this effort by developing and operating software that controls over 5,000 power converters, which supply current to the magnets in CERN's accelerators. As the software landscape of this team evolves, robust testing mechanisms are essential to validate existing functionalities, confirm new features, and ensure stability during code refactoring. Furthermore, the value of developing extensive test suites is greatly diminished without an effortless system for managing and reviewing them. To address this challenge, the FGC Test Manager was developed. This tool allows the developers within our team to efficiently manage their testing workflows.

Terminology

The following terms are defined to ensure clarity and consistency in the context of the FGC Test Manager tool.

- **Test Script**: A software script containing test cases to validate the functionality of another software system
- **Resource**: A collection of physical or virtual devices on which test scripts can be executed
- **Pair**: A combination of one test script and one resource that has been assigned to each other for testing purposes
- **Test**: The execution of a test script on a resource. A test is distinct from a test script; it represents the actual process of running the test script for a specific resource

Requirements for the FGC Test Manager

The following requirements outline the functionality and usability goals for the FGC Test Manager:

- Users must see a summary of the latest test results immediately upon opening the application.
- Displayed information must update in real-time.
- Users can filter the displayed information.
- Users can view data grouped by test scripts or resources.
- Users can toggle test inclusion in the nightly schedule and run tests on demand.
- Users must have access to detailed test logs.

USER WORKFLOW

The following section explains how users can interact with the tool.

Reviewing Test Results

Test outcomes can be viewed at two levels of detail. For each pair, the interface displays up to the last 15 test results, allowing users to assess recent activity quickly. Possible test states are summarized in Table 1. Selecting a test status opens a detailed view that includes logs from both the GitLab CI job and the test execution.

Customizations

The application allows users to tailor the interface to their specific needs. Users are given different filtering options, allowing them to refine the displayed data based on test script names, resource names, schedules, and tags. Additionally, users can choose to organize the displayed data either by test scripts (Fig. 1) or by resources (Fig. 2), switching between these views using the tabs located at the top of the content section. These customization features ensure that users can concentrate on the information most relevant to their tasks.

Test Execution

While a test is executing, the play button of the respective pair is temporarily replaced by a spinner to signalize test execution. The involved resource is locked whenever a test is executed, making it unavailable for other tests. This is mirrored using a lock symbol instead of the play button for all pairs that use the locked resource. Resources are

* fvonalbedyll@gmail.com

ISBN: 978-3-95450-248-6 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2025-THPS122

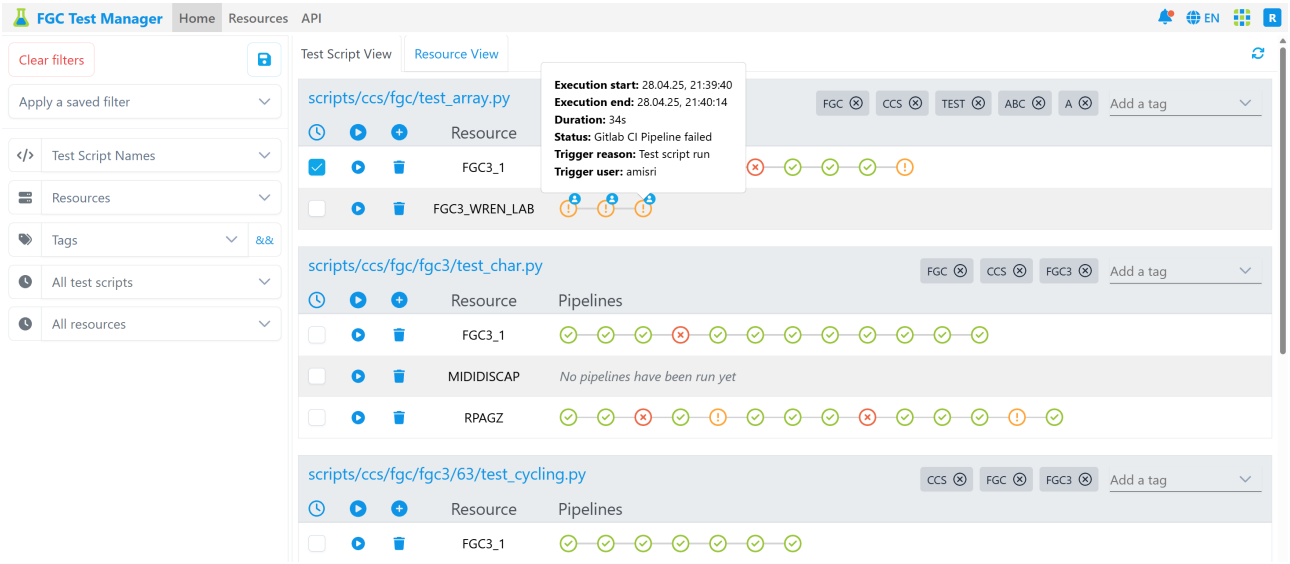

Figure 1: Example of the Test Script View.

locked at GitLab level using the *resource_group* property of jobs. This prevents parallel execution of jobs using the same resource and allows GitLab to handle orchestration based on the specified resource.

Scheduled Test Execution Schedules can be managed at three levels: individual pairs, test scripts, and resources. Each schedule determines whether the associated tests are included in the nightly scheduled Continuous Integration (CI) job. For a test to run as part of the nightly schedule, the schedules of all three elements—the pair, the test script, and the resource—must be enabled. This hierarchical scheduling ensures precise control over which tests are executed automatically.

On-demand Test Execution Users have the option to run tests on demand. This can be done for individual pairs, entire test scripts, or entire resources.

The application enables users to schedule recurring tests as part of the nightly CI job.

Assignments

Users can create a pair by assigning test scripts to resources or vice versa. This assignment is managed through a modal, which is opened by clicking the plus icon in the table of a test script or resource. The modal displays a list of all test scripts or resources that have not yet been assigned to the selected test script or resource, allowing users to create the pairing. Users can also delete assignments using the trash icon in each pair's respective row.

Test Script Management

Users can add a new test script or remove an existing one by committing the change to the fgc_tests repository. This action initiates an update, causing the test script changes to be displayed in the web application.

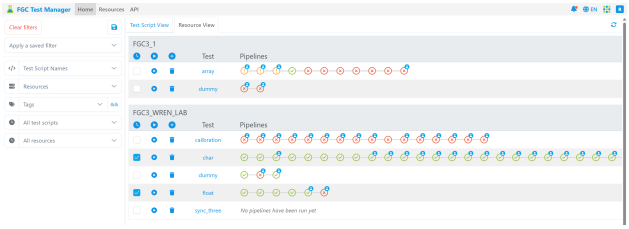

Figure 2: Example of the Resource View.

Resource Management

Resources can be added, removed, and modified via the *Resources* tab.

ARCHITECTURE

The architecture of the FGC Test Manager comprises four main components: the test suite repository (fgc_tests), a custom Python test framework (pyfgc_test_framework), the backend server, and the frontend client. Figure 3 offers a top-level perspective on how the system components interact.

Frontend

The front end is built using Vue3, a modern JavaScript framework that facilitates the development of reactive and dynamic user interfaces. It utilizes HTML for structuring content and CSS for styling and layout, ensuring a responsive and user-friendly design. Vue's component-based architec-ture allows for efficient code reuse and maintainability, while its reactive data binding ensures seamless updates to the in-terface as the testing workflow evolves. These technologies provide a robust foundation for creating an intuitive, adapt-able front-end experience.

fgc_tests repository

The fgc_tests repository is the storage location for test scripts and facilitates test execution using the GitLab CI/CD pipeline.

ISBN: 978-3-95450-248-6 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2025-THPS122

Table 1: Possible Test Outcomes

CI Job Finished	CI Job Failed	All Test Cases Passed	Status	Icon
n	-	-	Pending / Running	
y	y	-	Error	
y	n	n	Failure	
y	n	y	Success	

Script Storage When a developer commits a new test script to the repository or removes an existing one, a GitLab CI job triggers a call to the FGC Test Manager API. The FGC Test Manager server then updates the database with the necessary information about the updated test script. It then broadcasts this update to all its connected clients.

Test Execution Test execution is handled by a dedicated GitLab CI job runner. This runner is dynamically configured via environment variables:

- `TSTMGR_RESOURCE`: name of the resource
- `TSTMGR_DEVICES`: list of devices to run on
- `TSTMGR_SERVER_ENVIRONMENT_ID`: identifier of the instance requesting the test (`production` / `development`)
- `TSTMGR_SCRIPT_PATH`: path to the test script
- `TSTMGR_TRIGGER_USER`: username of the triggering user
- `TSTMGR_TRIGGER_REASON` (optional): user-provided comment for on-demand executions

Sending Test Changes Once a job has been created for a test, the FGC Test Manager API monitors its progress using a GitLab CI webhook. This webhook notifies the server whenever the status of a CI job changes. The server then updates the database with the new test status, providing real-time information to the front end for display.

Backend

The backend of the FGC Test Manager is implemented using FastAPI, serving as the system's core by managing data storage, facilitating communication with GitLab, and interacting with the front-end client. It ensures data consistency and smooth integration across all components.

The backend provides a comprehensive API that allows clients to access and update information about test scripts, resources, pairs, and test outcomes. It receives updates from GitLab, such as test job statuses or new artifacts, integrates this data into the database, and ensures clients always have an up-to-date view of the testing workflow.

Communication with the frontend ensures that clients receive requested data and are proactively notified of changes, keeping the interface accurate and responsive. For example, when a client creates a new pair assignment status, the backend updates the database and immediately broadcasts the update to all other clients.

By efficiently coordinating data flow and communication between GitLab and the frontend, the backend forms the backbone of the FGC Test Manager, ensuring reliability and responsiveness throughout the system.

pyfgc_test_framework

The pyfgc test framework is a Python-based library that provides a convenient and structured interface for interacting with Function Generation Controllers (FGCs). It offers streamlined access to FGC data. The framework simplifies tasks such as setting and retrieving property values, managing subscriptions, configuring protocols, and handling logs. It also includes utility functions for testing, configuration management, and automation, enabling users to efficiently perform tests, synchronize devices, and manage FGC states and logs within a robust and extensible testing environment.

CONCLUSION

The FGC Test Manager introduces a centralized solution for managing and executing software tests within the team. By simplifying workflows, it encourages more frequent testing and facilitates the inspection of results, promoting a culture of continuous improvement and software quality assurance. The FGC Test Manager represents a significant step forward in the team's efforts to ensure robust and reliable software systems for CERN's accelerator complex.

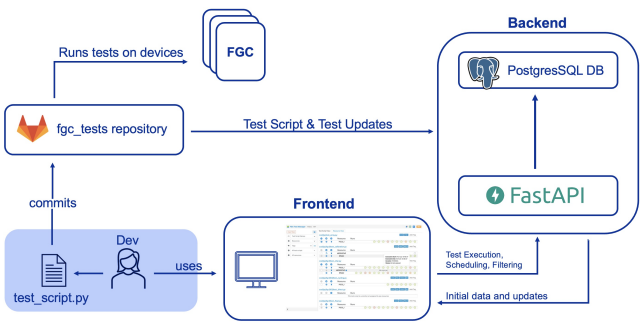

Figure 3: Overview of the main components and their interactions.