
facebook
INFRASTRUCTURE

Open Source Firmware Testing at Facebook

If you don’t test your firmware, your firmware fails you

Marco Guerri, Andrea Barberio

Production Engineers, Facebook

Agenda

- Problem statement
  - Why testing?
- Requirements
  - What about existing systems?
  - Enter ConTest
  - Scope
  - Use Cases
- Architecture
- Interfaces and plugins

Problem statement

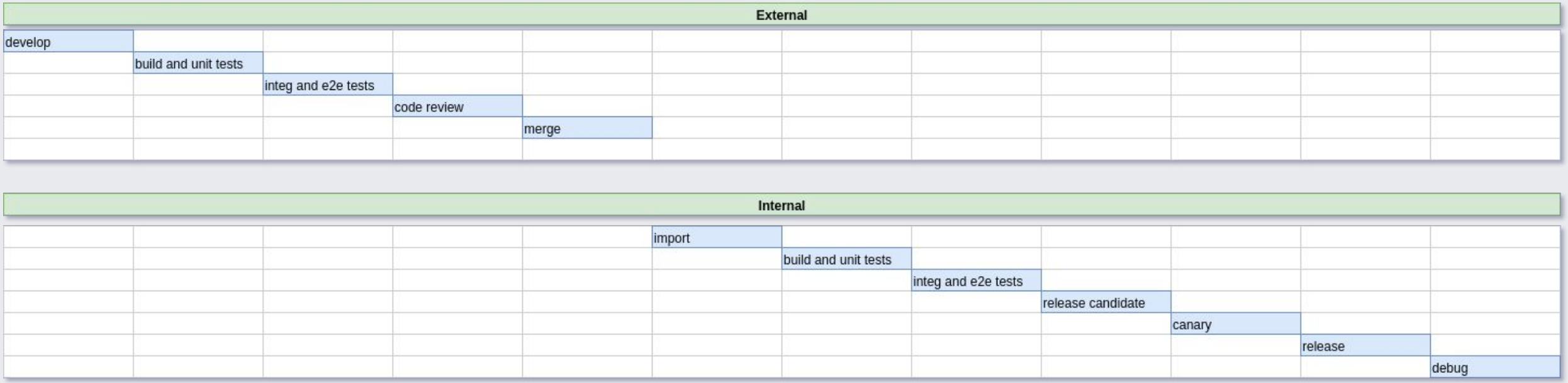
Problem statement

- We run open source system firmware in datacenter
- Development happens upstream on GitHub / Gerrit
- Simplified process:
  - develop
  - build and unit tests
  - integration and end-to-end tests
  - code review
  - release
  - debug

Problem statement

- We run open source system firmware in datacenter
- Development happens upstream on GitHub / Gerrit
- Simplified process:
  - develop
  - build and unit tests
  - **integration and end-to-end tests**
  - code review
  - release
  - debug

Development timeline



Why testing firmware?

- Pretty obvious with software. But firmware?

Why testing firmware?

- Pretty obvious with software. But firmware?
- Hardware is hard, firmware is not easy either

Why testing firmware?

- Pretty obvious with software. But firmware?
- Hardware is hard, firmware is not easy either
- Bugs can brick many devices. Reduced capacity

Why testing firmware?

- Pretty obvious with software. But firmware?
- Hardware is hard, firmware is not easy either
- Bugs can brick many devices. Reduced capacity
- Rolling out firmware takes longer than software

Why testing firmware?

- Pretty obvious with software. But firmware?
- Hardware is hard, firmware is not easy either
- Bugs can brick many devices. Reduced capacity
- Rolling out firmware takes longer than software
- Firmware influences machine’s behaviour and performance

Requirements

Requirements (1/2)

We want a firmware testing system that is

- **Open-source first**, and community-oriented. Together is better!

Requirements (1/2)

We want a firmware testing system that is

- **Open-source first**, and community-oriented. Together is better!
- **working with coreboot and LinuxBoot**

Requirements (1/2)

We want a firmware testing system that is

- **Open-source first**, and community-oriented. Together is better!
- **working with coreboot and LinuxBoot**
- **Robust**: minimize failures in prod, detect errors early

Requirements (1/2)

We want a firmware testing system that is

- **Open-source first**, and community-oriented. Together is better!
- **working with coreboot and LinuxBoot**
- **Robust**: minimize failures in prod, detect errors early
- **Generic**: can work in any infrastructure

Requirements (1/2)

We want a firmware testing system that is

- **Open-source first**, and community-oriented. Together is better!
- **working with coreboot and LinuxBoot**
- **Robust**: minimize failures in prod, detect errors early
- **Generic**: can work in any infrastructure
- **Scalable**: can run at datacenter scale

Requirements (2/2)

- **Human-friendly:**
  - **Simple by design**: easier to reason with, and to understand

Requirements (2/2)

- **Human-friendly:**
  - **Simple by design**: easier to reason with, and to understand
  - **Flexible**: assembled from independent components

Requirements (2/2)

- **Human-friendly:**
  - **Simple by design**: easier to reason with, and to understand
  - **Flexible**: assembled from independent components
  - **Easy to set up and maintain**: single binary, simple DB

Requirements (2/2)

- **Human-friendly:**
  - **Simple by design**: easier to reason with, and to understand
  - **Flexible**: assembled from independent components
  - **Easy to set up and maintain**: single binary, simple DB
  - **Easy to use**: configuration, not code

What about existing testing systems?

- We looked at several existing systems
- They didn’t satisfy all our requirements. Either:
  - complex to set up and maintain
  - complex to use
  - DUT-only test cases
  - too scoped functionalities

Enter ConTest

- **Con**tinuous and on-demand integration and e2e **Test**ing
- Single binary, plus SQL database
- Written in pure Go for ease and memory safety
- Can do much more than just firmware testing
- https://github.com/facebookincubator/contest

Scope

- What can ConTest do?
  - execute independent sequences of actions on DUTs
  - talk to external services
  - use jump hosts
  - microbenchmarks
  - report back customized results
  - operate in a single process on a Raspberry Pi, as well as in a datacenter-scale distributed infrastructure

Scope

- What ConTest **can't** do?
  - **no background processing:** only sequences of steps
  - **no dependency tree**: plugins implement independent execution flows
  - **no complex control flow** in the framework: just success/failure checks. Can be done in plugins

Use cases

Some of the possibilities include:

- validate network and local booting
- analyze boot messages from firmware and OS
- system stress testing
- performance analysis
- system provisioning validation
- run on-device actions
- orchestrate remote operations with external systems and services

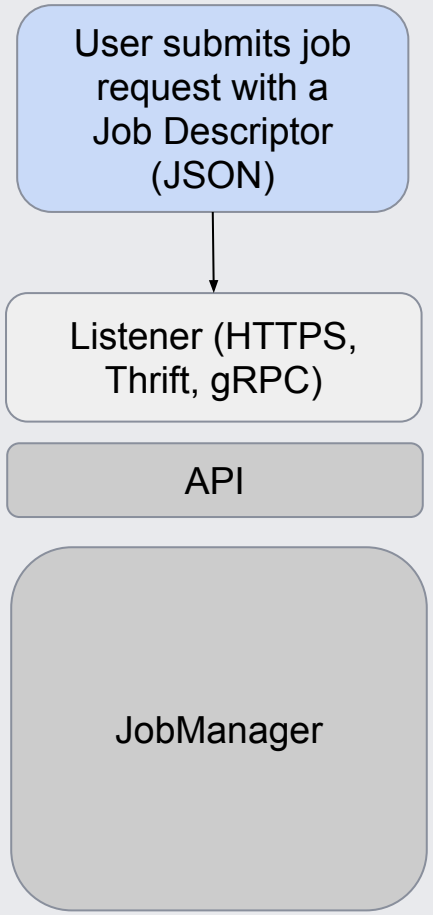
Architecture

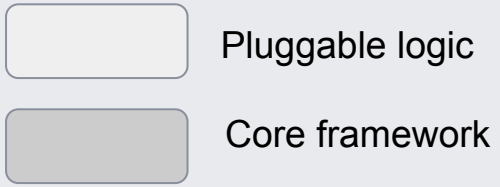
Architecture - Overview

Pluggable logic

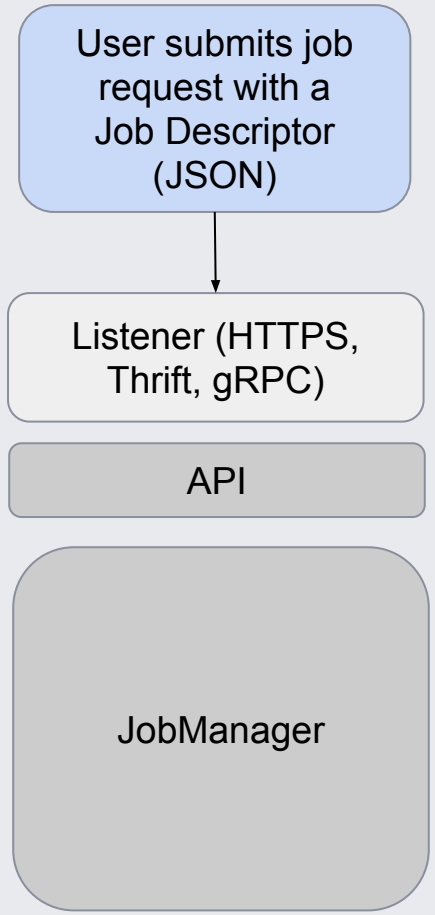
Core framework

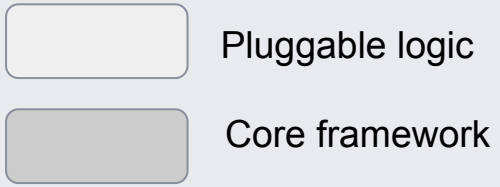
Architecture - Overview



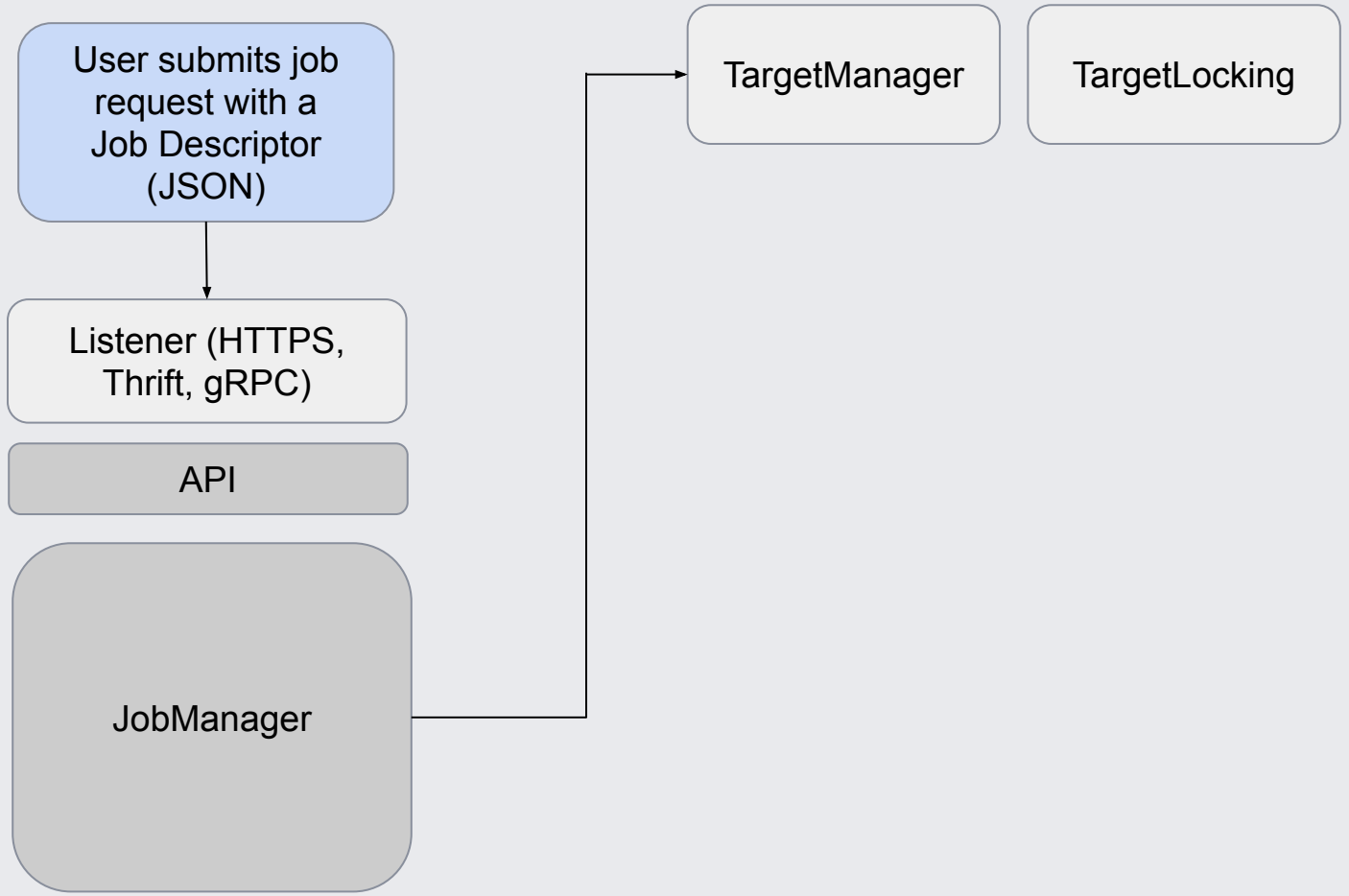


Architecture - Overview

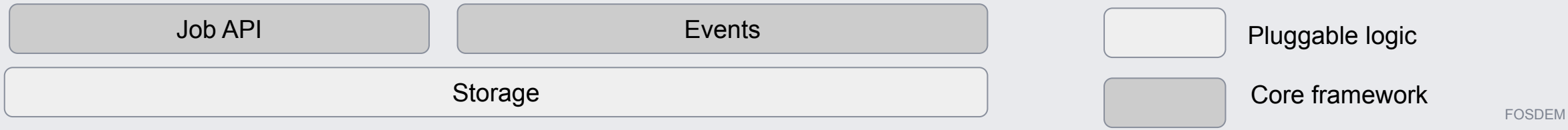




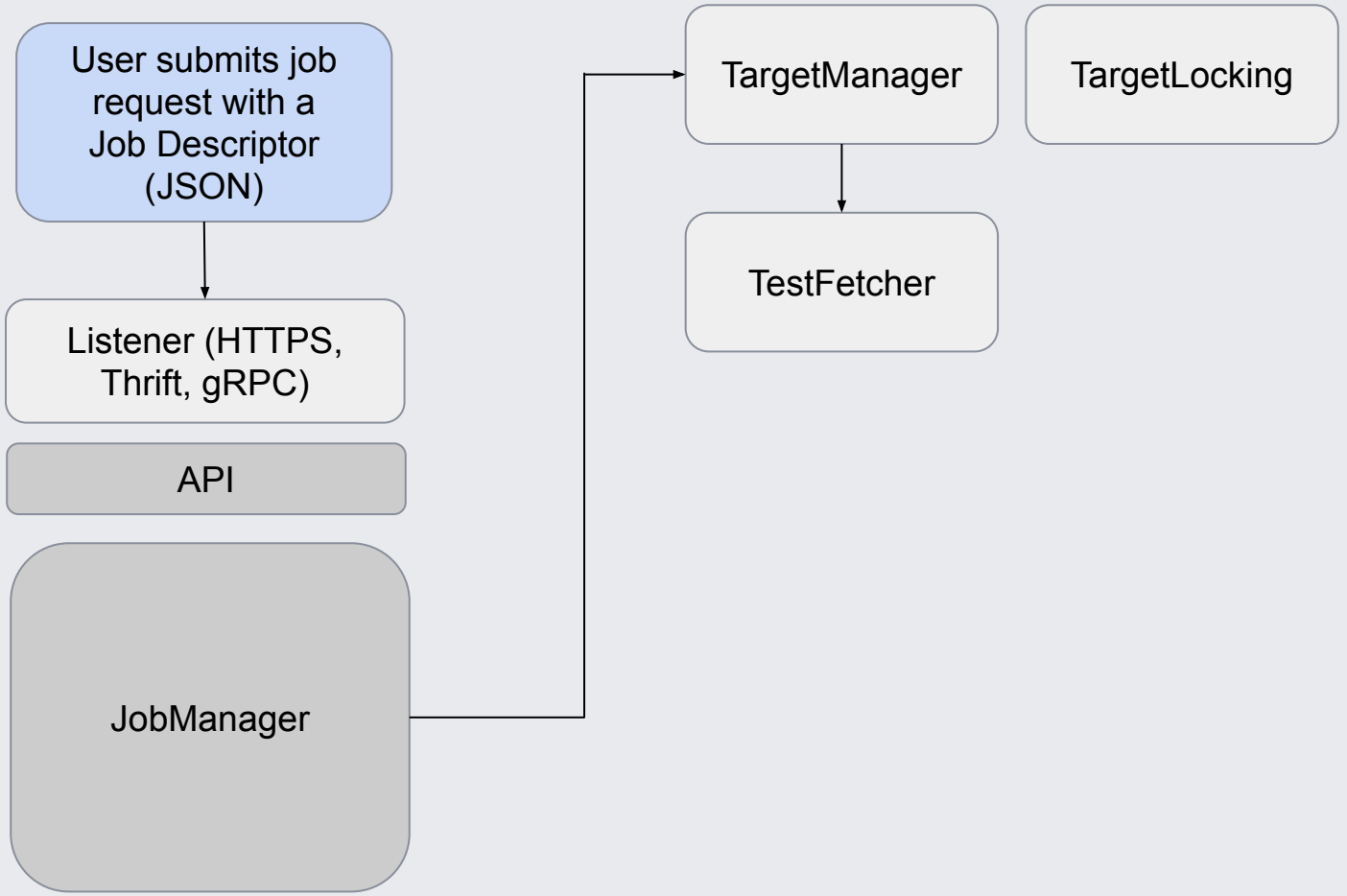
Architecture - Overview



Targets are fetched and locked.
The current ConTest instance has unique ownership of the Targets.



Architecture - Overview



Targets are fetched and locked.
The current ConTest instance has unique ownership of the Targets.

A description of the steps that constitute the test and associated parameters is fetched.

Job API

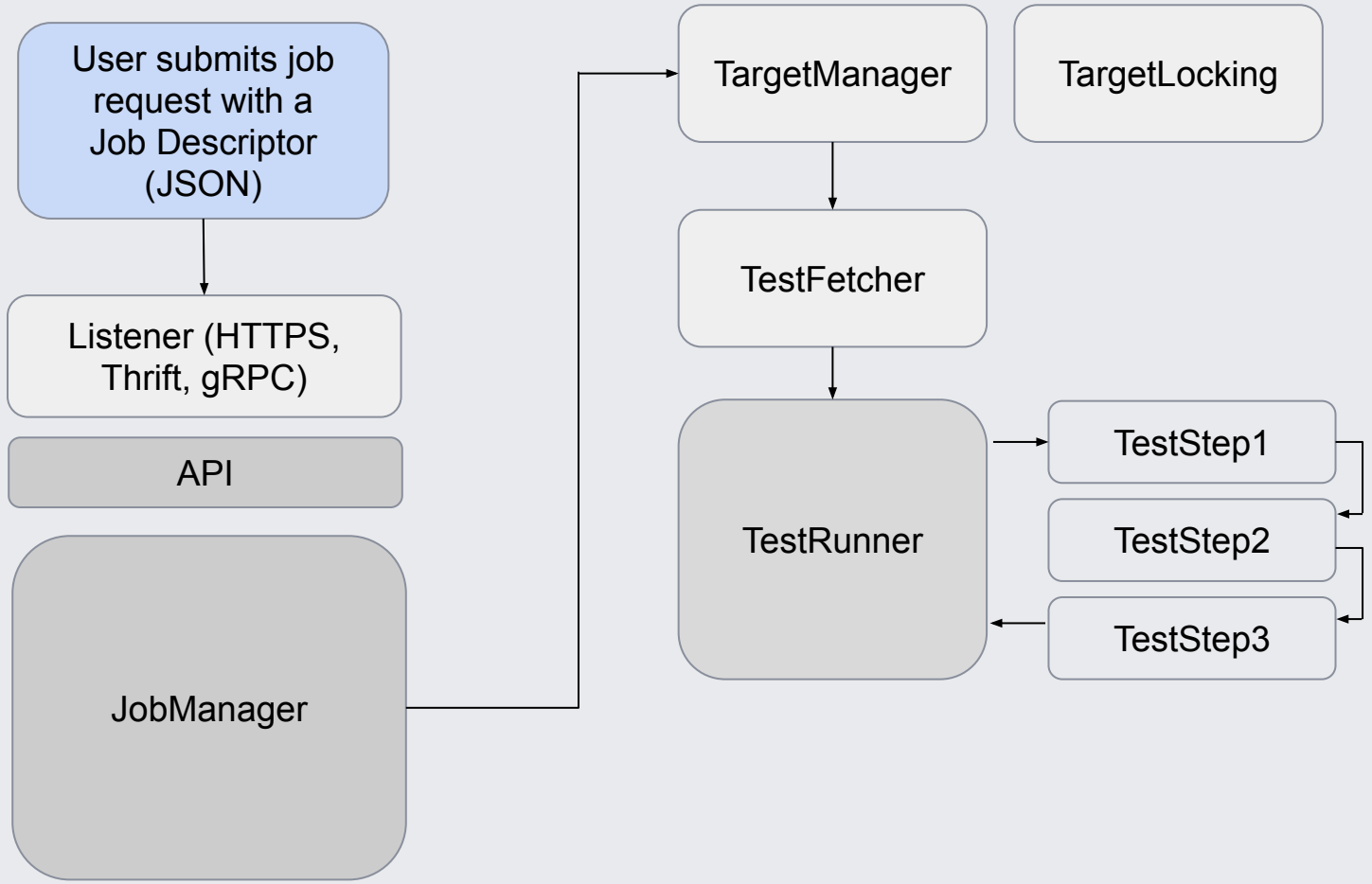
Events

Storage

Pluggable logic

Core framework

Architecture - Overview



Targets are fetched and locked. The current ConTest instance has unique ownership of the Targets.

A description of the steps that constitute the test and associated parameters is fetched.

Based on the description of the test, a pipeline is setup. The TestRunner orchestrates the flow of Targets through the various steps.

Job API

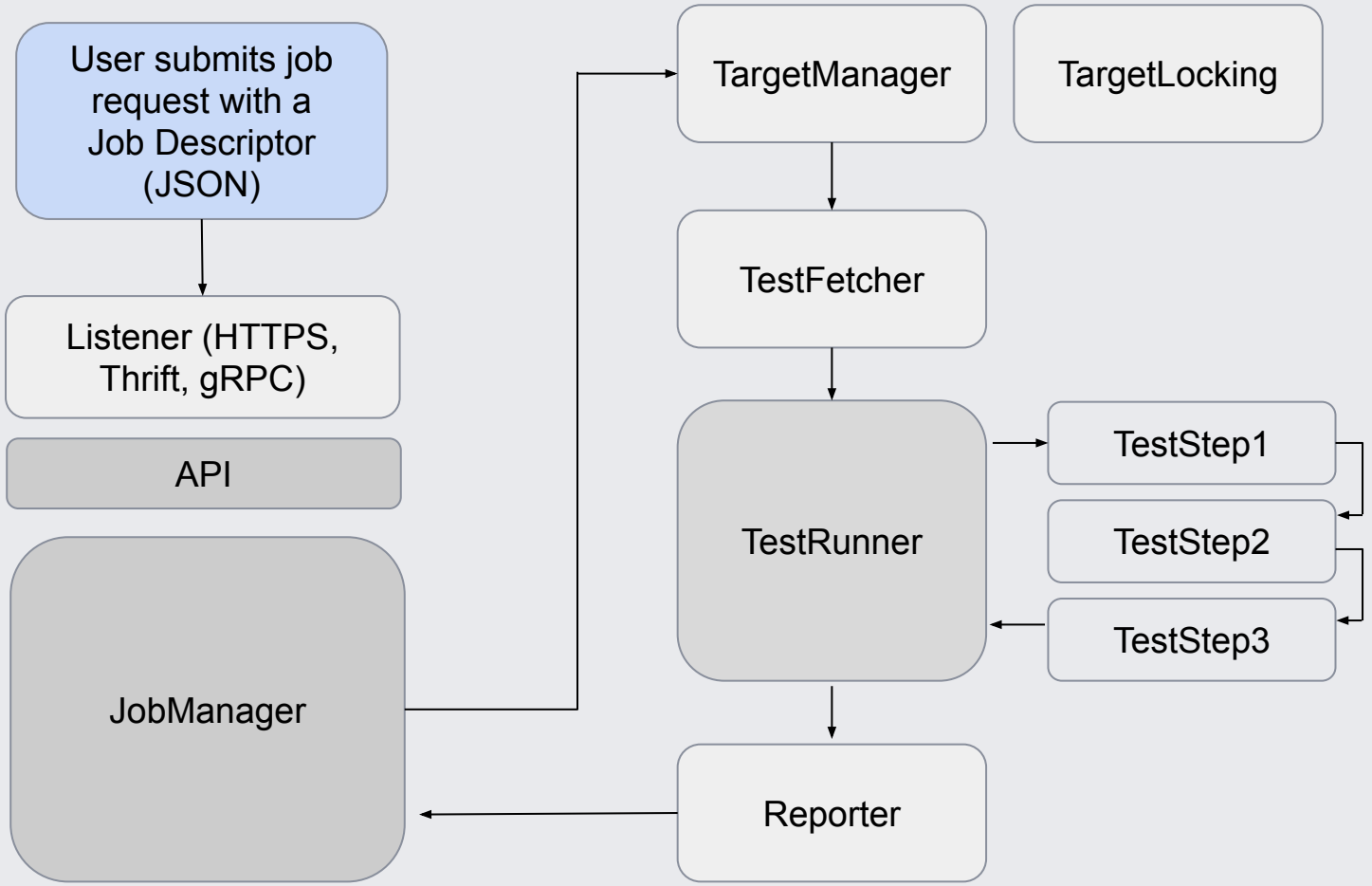
Events

Storage

Pluggable logic

Core framework

Architecture - Overview



Targets are fetched and locked.
The current ConTest instance has unique ownership of the Targets.

A description of the steps that constitute the test and associated parameters is fetched.

Based on the description of the test, a pipeline is setup. The TestRunner orchestrates the flow of Targets through the various steps.

A reporter is invoked to generate a custom description of the outcome of the test.

Job API

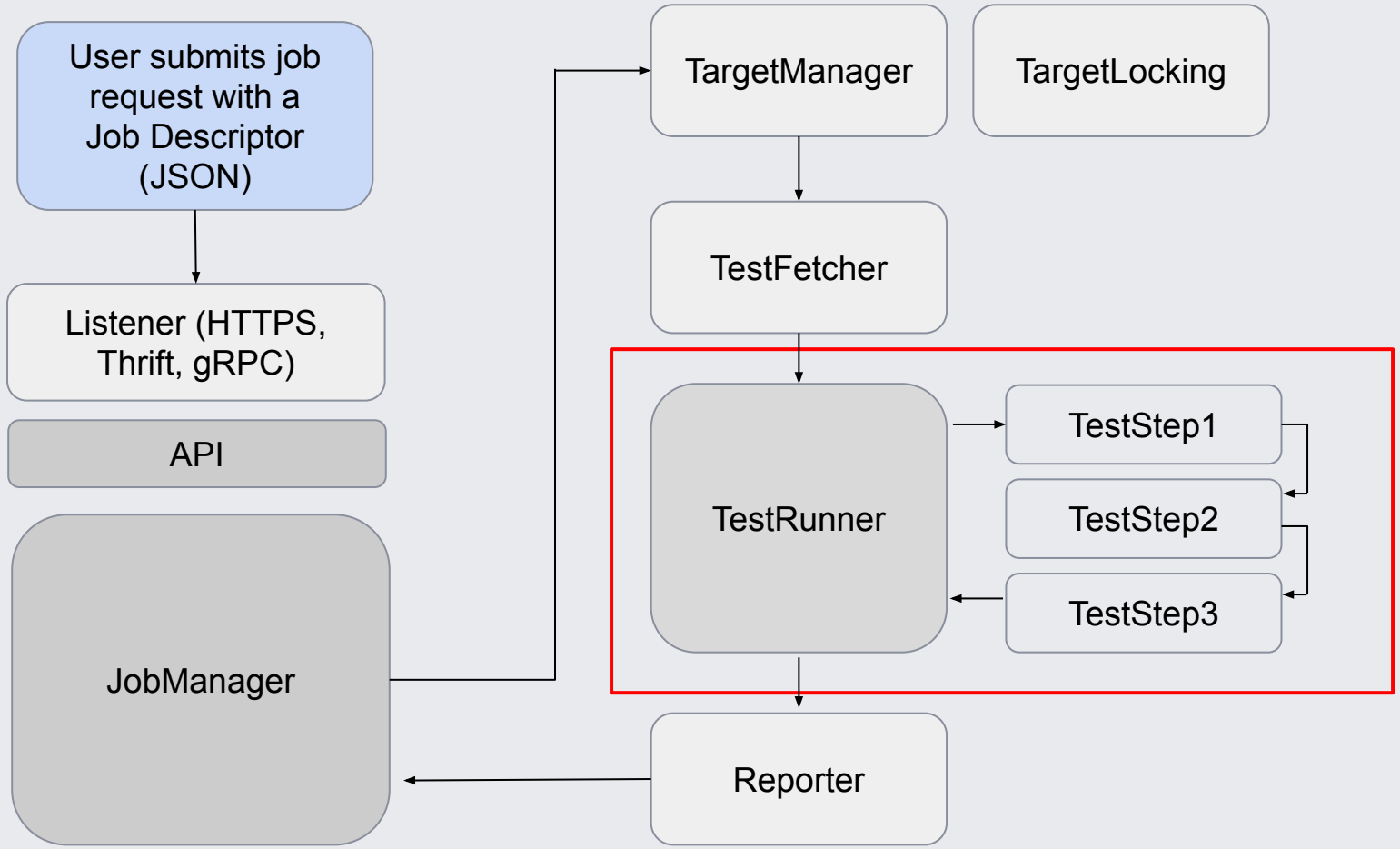
Events

Storage

Pluggable logic

Core framework

Architecture - Overview



Targets are fetched and locked. The current ConTest instance has unique ownership of the Targets.

A description of the steps that constitute the test and associated parameters is fetched.

Based on the description of the test, a pipeline is setup. The TestRunner orchestrates the flow of Targets through the various steps.

A reporter is invoked to generate a custom description of the outcome of the test.

Job API

Events

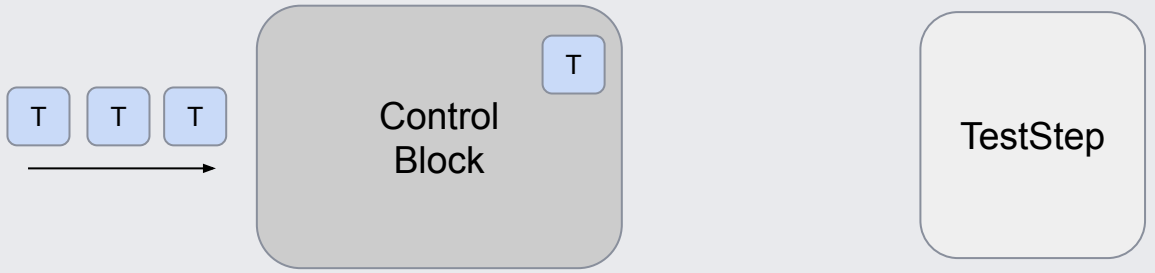
Storage

Pluggable logic

Core framework

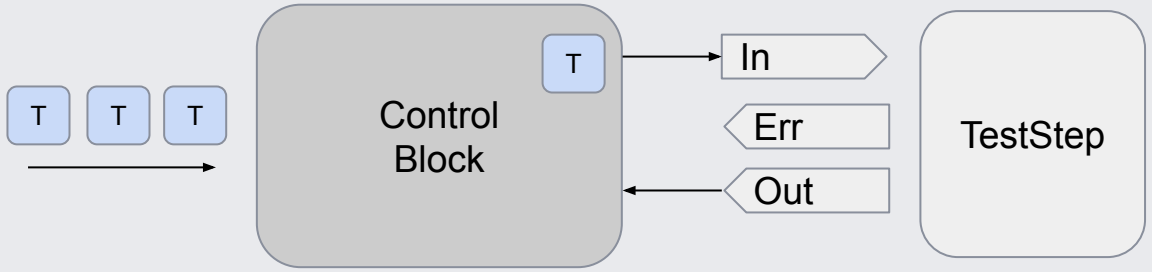
Architecture - Test Runner

The TestRunner controls the flow of Targets through the TestSteps.



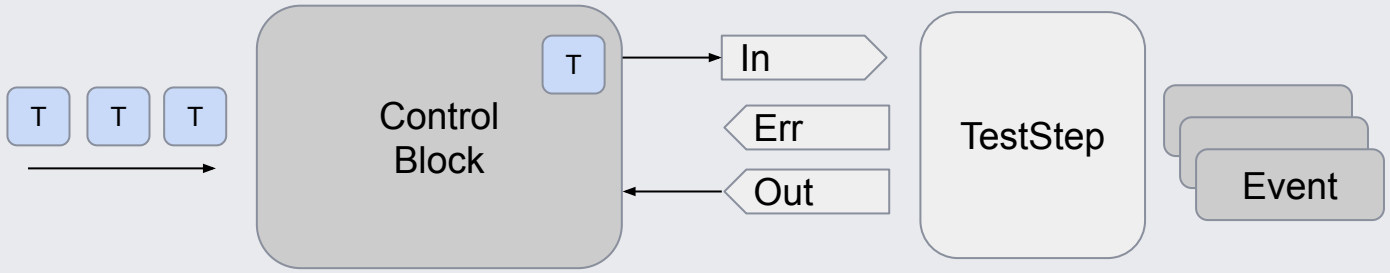
Architecture - Test Runner

The TestRunner controls the flow of Targets through the TestSteps.



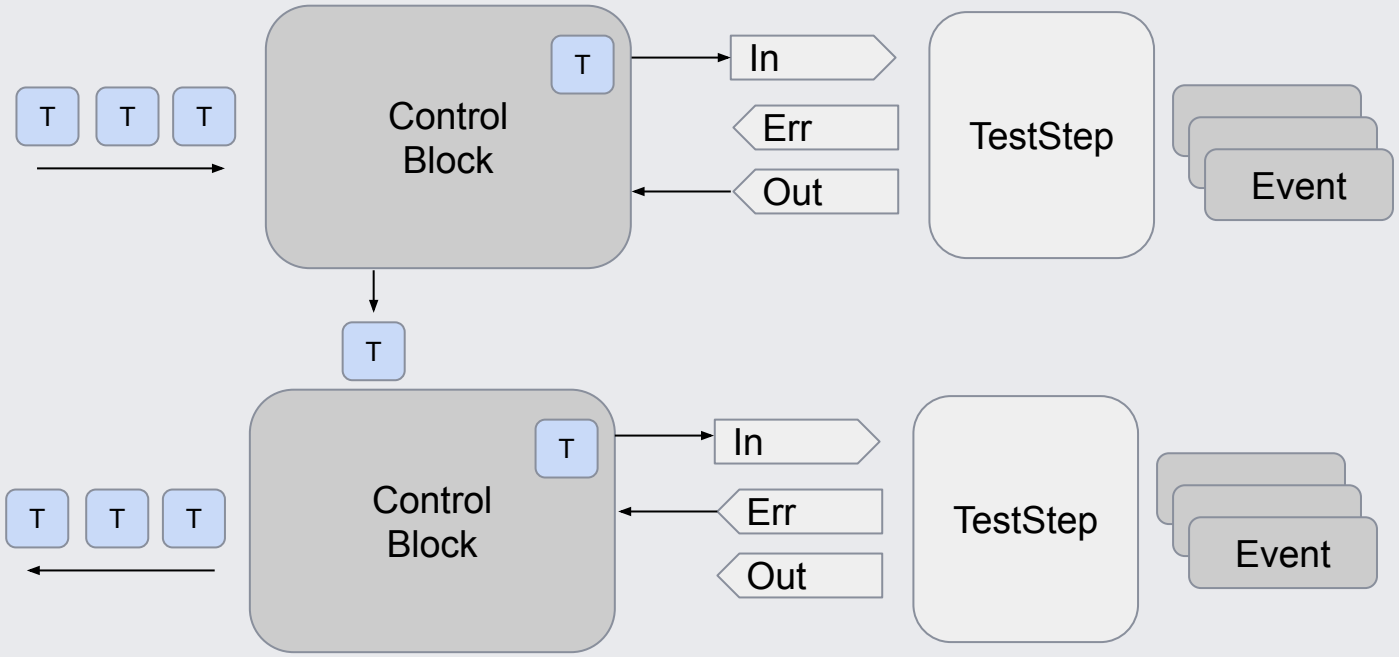
Architecture - Test Runner

The TestRunner controls the flow of Targets through the TestSteps.



Architecture - Test Runner

The TestRunner controls the flow of Targets through the TestSteps.

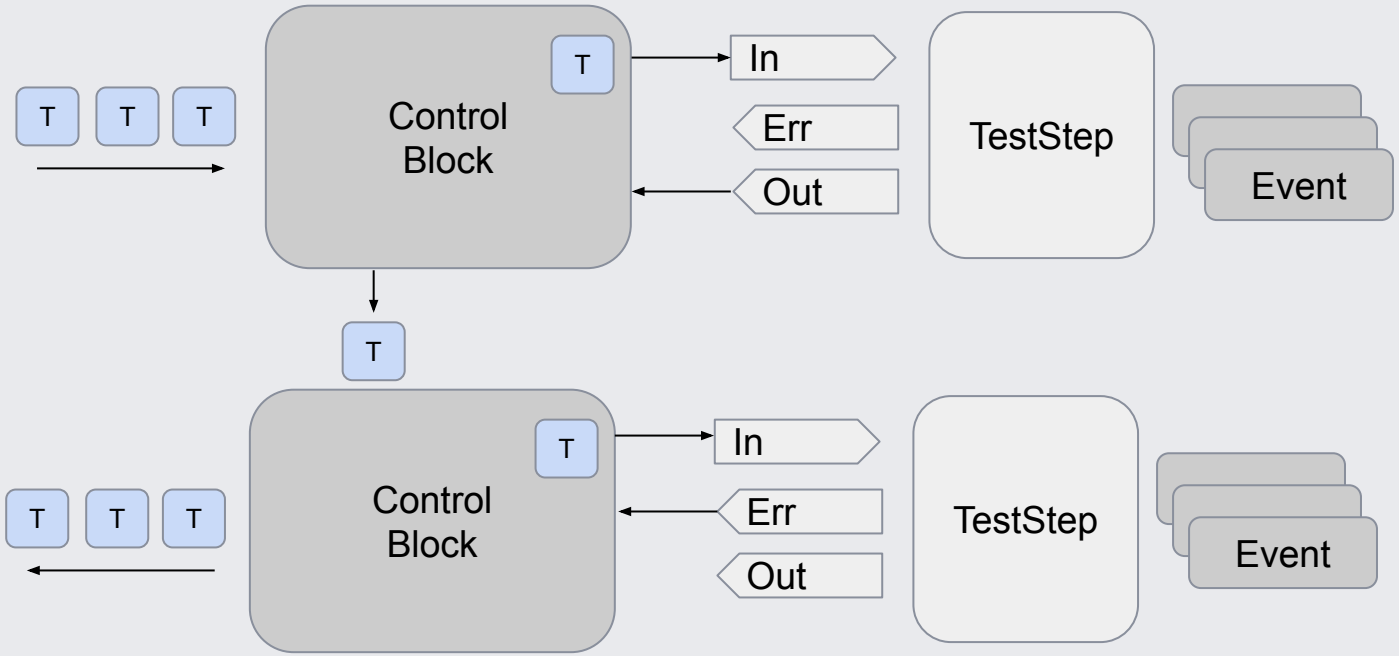


Architecture - Test Runner

The TestRunner controls the flow of Targets through the TestSteps.

A ControlBlock is associated to each TestStep to monitor the behavior of the plugin:

- Records success or failure of a Target via out and err channels
- Records Targets ingress and egress timestamps
- Enforces that targets fed to the TestStep must be returned in output
- Enforces that targets fed in input must be accepted with a timeout



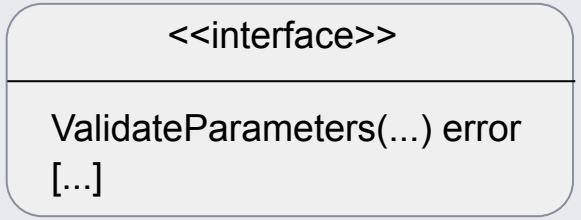
Interfaces and plugins

Interfaces

- Plugins must implement interfaces and meet requirements for I/O on channels, return values, timeouts, etc.
  - ConTest enforces that a job is terminated when a plugin does not comply with the requirements

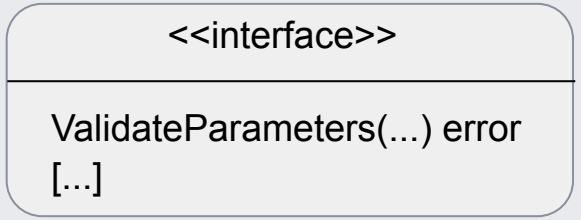
Interfaces

- Plugins must implement interfaces and meet requirements for I/O on channels, return values, timeouts, etc.
  - ConTest enforces that a job is terminated when a plugin does not comply with the requirements
- Interfaces are designed to allow for early validation of parameters



Interfaces

- Plugins must implement interfaces and meet requirements for I/O on channels, return values, timeouts, etc.
  - ConTest enforces that a job is terminated when a plugin does not comply with the requirements
- Interfaces are designed to allow for early validation of parameters
- Components are easily swappable, integration tests can use custom components that validate the logic of the framework



facebook | Questions

facebook

Thank you


facebook