

CIP Development process (SM-1)

Table of contents

1. Introduction
 2. Scope
 3. SM-1 : Development process
 - Configuration management with change controls
 - Audit Logging
 - Requirements definition
 - CIP Design
 - CIP Implementation
 - Testing and validation
 - Review and approval
 - Life-cycle
-

Revision History

Revision No	Date	Change description	Author	Reviewed by
001	2022-12-20	Draft document of CIP development process	Sai Ashrith	Dinesh Kumar
002	2023-03-28	Removed package proposal flowchart harcopy and replaced it with a link	Sai Ashrith	Dinesh Kumar
003	2023-06-21	Add the stages of development process as seperate sections	Sai Ashrith	Dinesh Kumar

Introduction

This document explains the development process used in CIP based on documented requirement checklist for SM-1 requirement in 62443-4-1. Design details such as system's devices and subsystems connections, exploitable areas and trust boundaries based on SD-1 requirement in 62443-4-1 are also included in this document.

Scope

This document explains the requirements which are clearly confirmed after discussion with Certification body. Some part of documentation is still not available due to need of further discussion with certification body such as configuration management document is SM-1.

SM-1 : Development process

The documentations and details required to fulfill SM-1 requirement in CIP are mentioned below :

Configuration Management with change controls Configuration management document is available [here](#). It has the details of versioning rules followed for CIP-Core and CIP-kernel.

Audit logging Audit logging procedures implemented in CIP and the measures taken to counter audit log failures are mentioned in this [document](#).

Requirements definition The overview of CIP requirements details are mentioned [here](#) as project goals.

Some of the project goals of CIP are :-

1. Super Long Term Support (SLTS) for **CIP kernel** and **CIP Core** packages up to 10+ years.
2. Regular testing of security updates on [CIP reference](#) boards.
3. Provide CIP reference images to CIP users.
4. Each workgroup to have it's own focus areas.

CIP members made this [document](#) where the functional and non-functional requirements are listed.

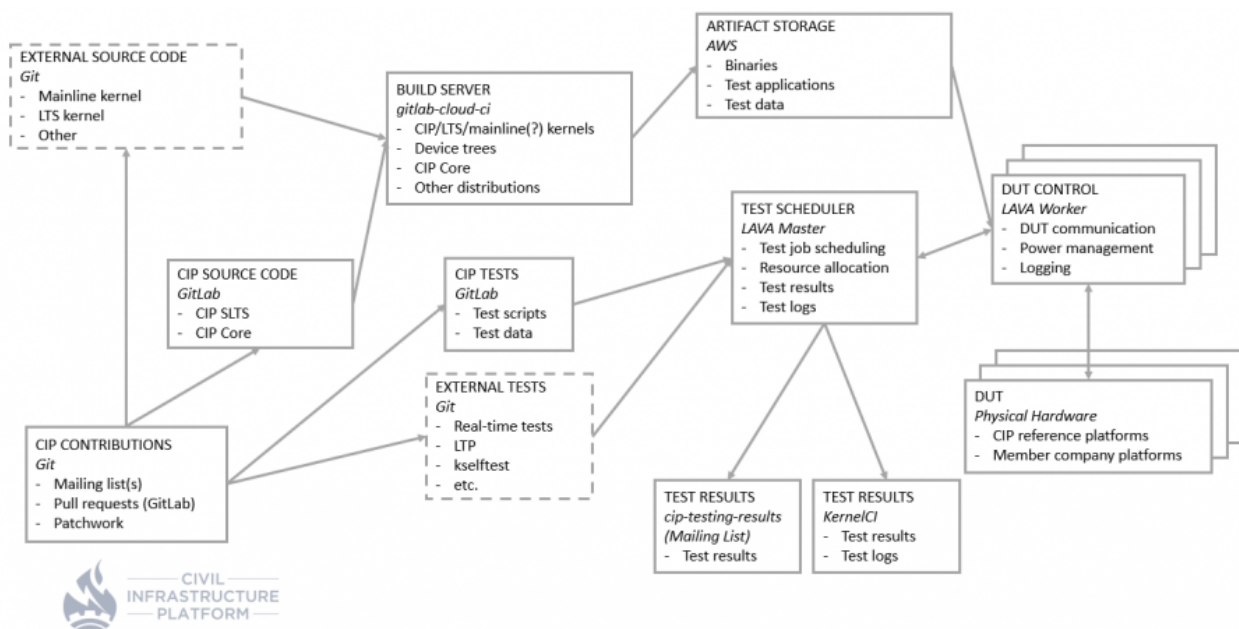
CIP Design **CIP-Core** re-uses **Debian** pre-built packages along with CIP-kernel which re-uses mainline kernel. The design and developments process which takes place in mainline kernel is mentioned [here](#). CIP-Core design and development details are mentioned in this [CIP-Core Wiki page](#) along with the [list of packages](#) being used in the system.

Every package used from Debian in CIP-Core follows the process as mentioned in this [illustration](#) before including in the design.

CIP Implementation **CIP-Core** implementation details are present in [CIP-Core Wiki page](#) and the [isar-cip-core](#) repository is actively maintained by the CIP developers. CIP kernel is actively maintained by the CIP developers in [CIP Kernel](#) repository. CIP kernel mostly re-uses mainline Linux kernel, so here is the [reference](#) with Linux kernel implementation details.

Here is an illustration which explains CIP-Core implementation :

Testing and validation The Centralized testing in CIP developers can run tests without having support to a growing list of reference platforms. Here is an illustration of testing overview in CIP.



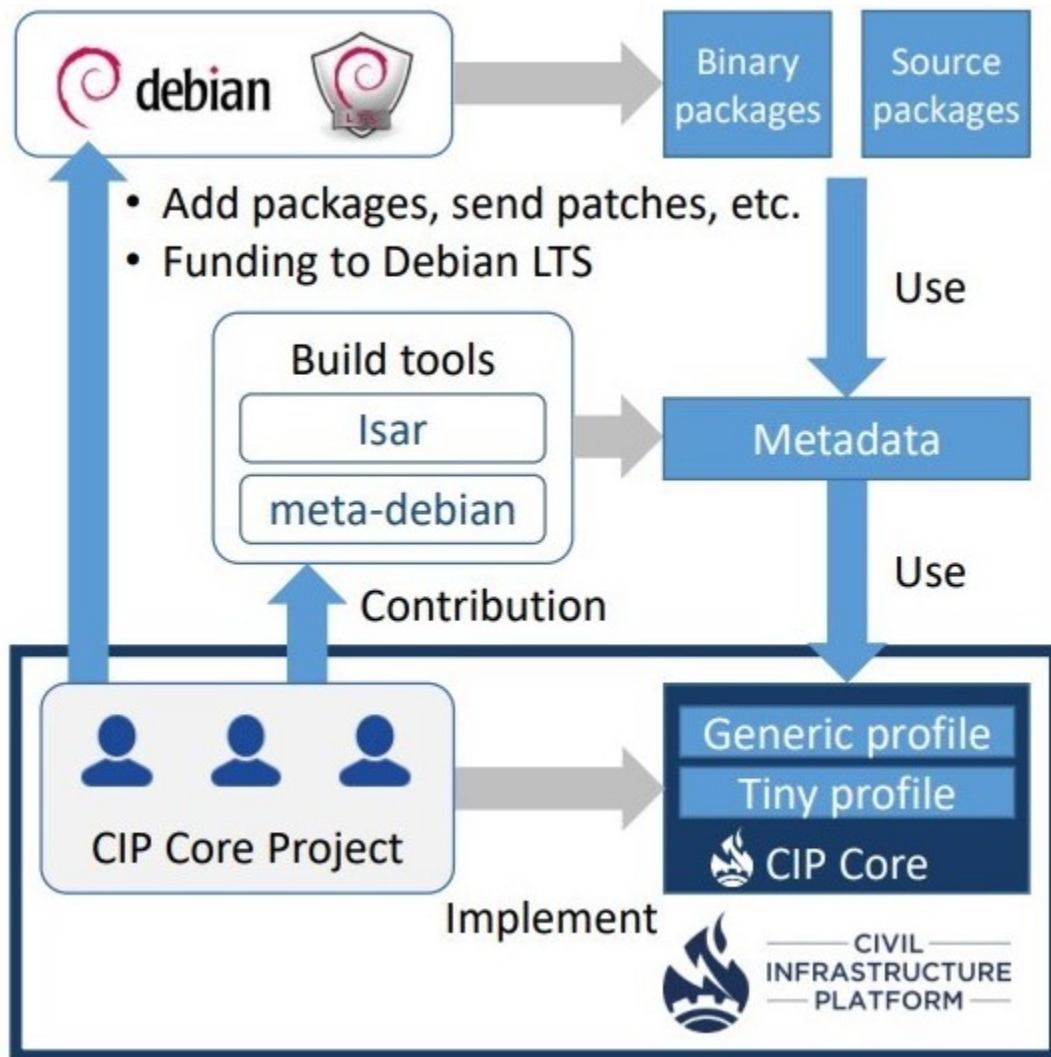


Figure 1: CIP-Core implementation

The continuous integration testing in CIP uses LAVA which helps to automatically test the updated software on CIP hardware. Kernel tests such as **Health and Meltdown checker, LTP tests and some real time tests will run in the LAVA.

In the context of process for repeatable testing, * The detailed process of submitting a custom LAVA job is mentioned [here](#). * The process of running IEC layer tests on the CIP security image is mentioned [here](#).

CIP-Core relies on upstream testing as it re-uses Debian packages without any modifications. The rigorous testing process done by Debian is mentioned at [Debian Testing page](#). Debian also has it's own CI for testing Debian packages and the results of some packages are available [here](#).

Review and approval This info needs to be discussed among CIP members and then should be documented.

Life-cycle The life-cycle details of **CIP-Core**, **CIP-Kernel** and their upstream projects are mentioned in [cip-lifecycle](#) repository.