Testing gaps - core components

Contents

General considerations 2
apertis-update-manager 3
apparmor 4
busybox 4
dbus 5
gnutls28 6
linux-image 6
openssl 7
ostree 8
rust-coreutils 9
rust-findutils 9
rust-sequoia-sqv 10
systemd 11
Summary of proposals 11

General considerations

As described in Apertis test strategy[1] the approach to do gap analysis is to classify the components under different categories and based on the expected levels of testing for each of them provide a report about the gaps.

In general, based on the current workflow, most of the component already meet some standard level of testing, and share some common status which is described below.

As a general idea, testing should focus in Apertis specific components and components with delta from Debian. For components not under heavy development in Apertis the focus should be on integration tests.

Please refer to the Apertis test strategy[2] for more details on the loops described below.

Local loop

Developer tests: Required, ad hoc during development.

Unit tests: Components with unit test support in Debian inherit this property, each component will be analyzed individually, below.

CI loop

Linters: Linters are a nice to have feature, but only encouraged for component under development in Apertis.

License scan: License scan is already triggered for all the branches in all the packages. This scan is meant to provide a full copyright report available and to

[1] https://www.apertis.org/concepts/distribution/test-strategy/
[2] https://www.apertis.org/concepts/distribution/test-strategy/#loops-and-types

raise a warning in case a license does not match Apertis license expectations[3].

OBS build: OBS build is already run in WIP branches/MRs for all the packages.

Integration tests: This type of test helps to ensure that proposed changes will not affect the stability of the final solution. Adding this kind of test can make a huge impact by catching regressions earlier, making components under development or with important delta from Debian good candidates. Currently, no package runs integration tests.

Review loop

Review is always required for any kind of change.

Image loop

Installation: Core components are part of daily images and installed on them.

License compliance: Core components are already part of daily images and checked as part of the license checks.

Orchestrator loop

Installation: Core components are already part of daily images and installed on them.

Integration tests

Functional tests: This type of test is analyzed on a per component basis.

Performance: For the purpose of Apertis as distribution performance is evaluated in a high level approach setting time constrains in the functional tests to run, to spot deviation from expected performance behavior.

apertis-update-manager

Category	Classification
Source	1 - Apertis specific component
Activity	1 - Minimal upstream activity
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests not available. A minimal set of test is required to validate that the basic functionality works after changes during development, such as read/write boot count from NVME and UEFI.

[3] https://www.apertis.org/policies/license-expectations/

CI loop

Linters: Linters should be added to ensure all the contributors make changes using the same standards.

Integration tests: Since this component is Apertis specific, and has low community activity, the currently available integration tests are already run as part of CI.

Integration tests

Functional tests: Functional tests are already available, currently 18 test suites run. AUM is one of the most exercised components in all the available architectures, including amd64 after adding support to the new reference board.

apparmor

Category	Classification
Source	2 - Significant delta from Debian
Activity	2 - Medium upstream activity
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests already available, currently 71607 tests run. Since this package is not under heavy development improving the coverage is not required.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Since this component has a significant delta from Debian, the currently available integration tests are already run as part of CI.

Integration tests

Functional tests: Functional tests are already available, currently 14 specific test suites run. The Apparmor is one of the most exercised components in all the available architectures. This component is a special one, since it has to deal with security and is tightly coupled to a desired security configuration. For this reason it is recommended to review periodically the functional tests and adapt them to the needs of downstream distributions and product teams.

busybox

Category	Classification
Source	2 - Delta from Debian
Activity	2 - Medium upstream activity
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests already available, currently 714 tests run. Since this package is not under heavy development improving the coverage is not required.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Recommended given this component has a delta from Debian, is used during images bootstrap, as well as a partial replacement for **coreutils**, where changes may affect core functionality. In this context, testing bootstrapping and image creation is a recommended step.

Integration tests

Functional tests: Apertis uses busybox to provide support for some **coreutils** utilities such as diff, grep and sed, which are used by the standard Debian utilities to handle packages and also used by tests scripts of different components. Based on these facts the basic integration tests are already satisfied by the image generation, for this reason, additional tests are not required.

dbus

Category	Classification
Source	3 - No delta from Debian
Activity	1 - High upstream activity
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests already available, currently around 1986 tests run. Since this package is not under heavy development, improving the coverage is not required.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Not required since this component has high upstream activity/support, there is no delta from Debian and it is not under development.

Integration tests

Functional tests: Apertis already runs the upstream supported **dbus** on daily images. Also, many Apertis components use **dbus** to exchange messages, from this perspective and since this component does not contain any delta from Debian, adding more integration tests is not required.

gnutls28

Category	Classification
Source	3 - No delta from Debian
Activity	2 - Medium upstream activity
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests already available, currently 746. Since this package is not under heavy development improving the coverage is not required.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Not required since this component has upstream support, there is no delta from Debian and it is not under development.

Integration tests

Functional tests: Several Apertis components use **gnutls**, as in Debian, however, since it does not contain any delta from Debian, integration tests for itself are not strictly required.

linux-image

Category	Classification
Source	2 - Latest LTS version
Activity	1 - High upstream activity
Commonality	1 - High commonality

Category	Classification
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Running unit test on linux is very difficult since it is a very low level package and very dependent on the hardware. Also since this package is not under development adding unit tests is not required.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Linux is a special component as it is very tight to the hardware and Apertis ships the latest LTS version available for each release. Even if it is not formally under development in Apertis, configurations are customized and drivers added to support the reference boards. Given that, the tests to verify that booting works as expected are already run as part of CI.

Integration tests

Functional tests: Linux is a core component which is tested in every reference board available to make sure the basic functionality it provides works as expected. Since different boards use different hardware and drivers, the different combinations should be tested.

These tests are usually tight to high level components which make use of the functionality. For instance **connman** is the service used to handle network connections, which makes use of Linux to access the network interfaces. With this idea in mind integration tests should be divided taking into account tests for high level components:

- networking: connman
- bluetooth: bluez
- audio: pipewire/gstreamer
- video: pipewire/gstreamer

For basic functionality, like booting, access media devices such as SD card test are already available.

As this component is likely to be customized by downstream distributions, product teams should pay special care and adjust tests according to their needs.

openssl

Category	Classification
Source	2 - Replacement of gnutls
Activity	2 - Medium upstream activity
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests already available, currently 2622 tests run. Since this package is not under development improving the coverage is not required.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Several components use **openssl** instead of **gnutls** due to license restrictions, causing a delta from Debian. For this reason, the currently available integration tests are already run as part of CI.

Integration tests

Functional tests: Several Apertis components use **openssl** instead of **gnutls** due to license restrictions, for this reason integration tests for the components with delta from Debian are recommended. For that, tests based on glib-networking, which uses openssl, are already available and should cover potential regressions.

ostree

Category	Classification
Source	2 - Delta from Debian
Activity	2 - Medium upstream activity
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests already available, currently 242 which cover the most common uses.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Running integration tests as part of CI is recommended since this package has some delta from Debian and provides critical functionality.

Integration tests

Functional tests: OSTree is tested while testing **Apertis Update Manager** which is the high level application that handles upgrades in Apertis covering the common scenarios, which already has a good testing coverage as previously mentioned.

rust-coreutils

Category	Classification
Source	2 - Replacement of coreutils
Activity	2 - Medium upstream activity
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests are already available, currently 1477 are available but not enabled. The recommendation is to enable them. It is important to note that the support for unit tests in **debcargo** is under development and from the current set of unit tests 1 fails and 27 are ignored.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Recommended since this package has a delta from Debian, is used during images bootstrap and is a replacement for **coreutils**, where changes may affect core functionality. In this context, testing bootstrapping and image creation is a recommended step.

Integration tests

Functional tests: The basic use of rust-coreutils functionality is indirectly tested by the use of them at different stages, such as image generation and testing scripts. For this reason, additional tests are not required.

rust-findutils

Category	Classification
Source	2 - Replacement of findutils
Activity	2 - Medium upstream activity

Category	Classification
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests are already available, currently 151 tests are available, but not enabled. The recommendation is to enable them. It is important to note that the support for unit tests in **debcargo** is under development.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Recommended since this package has a delta from Debian, and it is used as replacement from **findutils**. In this context, testing bootstrapping and image creation is a recommended step.

Integration tests

Functional tests: The basic use of **rust-findutils** functionality is indirectly tested by the use of them at different stages, such as image generation and testing scripts. For this reason, additional tests are not required.

rust-sequoia-sqv

Category	Classification
Source	2 - Replacement of gpgv
Activity	2 - Medium upstream activity
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests are not yet available, however, since this component is not under development adding them is not required.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Recommended since this package has a delta from Debian and it is used as replacement from **gpgv**. In this context, testing bootstrapping and image creation is a recommended step.

Integration tests

Functional tests: The basic use of rust-sequoia-sqv functionality is indirectly tested by the use of it at different stages, such as image generation and testing scripts. For this reason, additional tests are not required.

systemd

Category	Classification
Source	2 - Delta from Debian
Activity	1 - High upstream activity
Commonality	1 - High commonality
Criticality	1 - High criticality
Target	1 - Use in target devices

Local loop

Unit tests: Unit tests already available, currently 596, which cover the most common uses.

CI loop

Linters: Linters are not required since this component is not under development inside the Apertis project.

Integration tests: Since this component has some delta from Debian, the current integration tests to verify that booting works as expected are already run as part of CI.

Integration tests

Functional tests: The standard use of **systemd** is tested indirectly by the use of it at different stages, such as booting images and test running services such as **connman** and **pipewire**. Since the delta is more focused in avoiding bashisim additional tests are not required.

Summary of proposals

Component	Test on MR	Bootstrap on MR	Other tests	Comments
apertis-update-manager		Not required	Add unit tests, linters	
apparmor		Not required		Review for downstrean
busybox	Not required	Need to add		
dbus	Not required	Not required		
gnutls28	Not required	Not required		
linux-image		Not required		Check high level applic
openssl		Not required		

Component	Test on MR	Bootstrap on MR	Other tests	Comments
rust-coreutils	Not required	Need to add	Enable unit tests	
rust-findutils	Not required	Need to add	Enable unit tests	
rust-sequoia-sqv	Not required	Need to add	Enable unit tests	
systemd		Need to add		