Low-Level API Test Plan

v.1.0.0

Important notices, IPR statement, disclaimer and Copy-right

This chapter contains important information about PRPL and this document (hereinafter ‘This PRPL Document’).

ABOUT PRPL

The prpl Foundation (PRPL) is a not-for-profit organization which publishes documents including, but not limited to, Requirements, Specifications, Recommendations, API application programming interfaces, and Test Plans.

THIS MAY NOT BE THE LATEST VERSION OF THIS PRPL DOCUMENT

This PRPL Document is the output of the Working Groups of the PRPL and its members as of the date of publication. Readers of This PRPL Document should be aware that it can be revised, edited or have its status changed according to the PRPL working procedures.

THERE IS NO WARRANTY PROVIDED WITH THIS PRPL DOCUMENT

The services, the content and the information in This PRPL Document are provided on an "as is" basis. PRPL, to the fullest extent permitted by law, disclaims all warranties, whether express, implied, statutory or otherwise, including but not limited to the implied warranties of merchantability, non-infringement of third-parties rights and fitness for a particular purpose. PRPL, its affiliates and licensors make no representations or warranties about the accuracy, completeness, security or timeliness of the content or information provided in the PRPL Document. No information obtained via the PRPL Document shall create any warranty not expressly stated by PRPL in these terms and conditions.

EXCLUSION OF LIABILITY

Any person holding a copyright in This PRPL Document, or any portion thereof, disclaims to the fullest extent permitted by law (a) any liability (including direct, indirect, special, or consequential damages under any legal theory) arising from or related to the use of or reliance upon This PRPL Document; and (b) any obligation to update or correct this technical report.

THIS PRPL DOCUMENT IS NOT BINDING ON PRPL NOR ITS MEMBER COMPANIES

This PRPL Document, though formally approved by the PRPL member companies, is not binding in any way upon the PRPL members.

Patents essential or potentially essential to the implementation of features described in This PRPL

INTELLECTUAL PROPERTY RIGHTS

Document may have been declared in conformance to the PRPL IP Policy (available at the PRPL website: www.prplFoundation.org).

COPYRIGHT PROVISIONS

INCORPORATING PRPL DOCUMENTS IN WHOLE OR PART WITHIN DOCUMENTS RELATED TO COMMERCIAL TENDERS

COPYING THIS PRPL DOCUMENT IN ITS ENTIRETY

Executive Summary

This Low-Level API (LL-API) test plan outlines the comprehensive validation strategy for a Hardware Under Test's (HUT) implementation of the prpl Low-Level API Specification. The testing approach focuses on verifying that SoC vendor BSPs and SDKs expose the standard Linux kernel interfaces mandated by the specification, and that these interfaces behave correctly under both normal operation and stress conditions.

By exercising the Low-Level API directly at the kernel and system-call layer, the test suite validates the core kernel subsystems on which higher-layer CPE software depends. This systematic approach ensures comprehensive coverage of the interfaces defined by the specification while proactively identifying defects, kernel instability, and version drift before they surface in the field.

Key objectives of this test plan include:

- Verification of Linux kernel, OpenWRT, prplMesh, prplLCM, OB-USP-A, wpa_supplicant, and hostapd versions
- Validation of core functions including reboot, factory reset, and CPU frequency and idle-time management
- Assessment of wireless capabilities and scan behavior via cfg80211/nl80211
- Evaluation of ethernet interface management and IPv4/IPv6 address, route, and MTU handling
- Testing of IPv4 multicast group management, source filters, and IGMP query processing
- Verification of kernel crypto API robustness under malformed inputs and boundary conditions
- Assessment of storage volume identification against the prplOS partition layout

This particular test plan is one component of a broader suite of test plans for formally Certifying implementations of "prplWare and High-Level API".

The Low-Level API is, by design, platform-specific: it standardizes how SoC vendors expose kernel subsystems to the software stacks above. Because these interfaces are tightly coupled to the underlying hardware and kernel, this test plan is one of the non-platform-agnostic components of the certification suite. Its goal is to discourage forking by verifying the authenticity and full realization of the prplWare components in the specific implementation under test.

This testing initiative will provide crucial validation of the standardized kernel interfaces that underpin prplWare deployments, ensuring the platform meets its intended operational requirements.

Revision History

Version	Date	Notes
1.0.0	September 2, 2026	• Initial release from Certification Technical Working Group.
0.0.2	January 14, 2026	• Added tests 1.6 and 1.7. • Added tests 2.16 to 2.35.
0.0.1	January 10, 2026	• Create document. • Added tests 1.1 to 1.5 for version checking. • Added tests 2.1 to 2.15 for CPU, Reboot, Wi-Fi, and IP tests.

Test Organization

This document organizes tests by group based on related test methodology or goals. Each group begins with a brief set of comments pertaining to all tests within that group. This is followed by a series of description blocks; each block describes a single test. The format of the description block is as follows:

Test Label	The Test Label and Title comprise the first line of the test block. The Test Label is composed of the short test suite name, the group number, and the test number within the group, separated by periods.
Purpose	The Purpose is a short statement describing what the test attempts to achieve. It is usually phrased as a simple assertion of the feature or capability to be tested.
Reference	The References section lists cross-references to the specifications and documentation that might be helpful in understanding and evaluating the test and results.
Functionality Tag	The functionality tag indicates whether the test is mandatory or optional as defined in prpl Certification Guide.
Procedure	The procedure indicates the steps, in order, taken to perform the test. Actions are steps taken by the test tool. Expected Behavior describes what a device under test must do to pass the test.

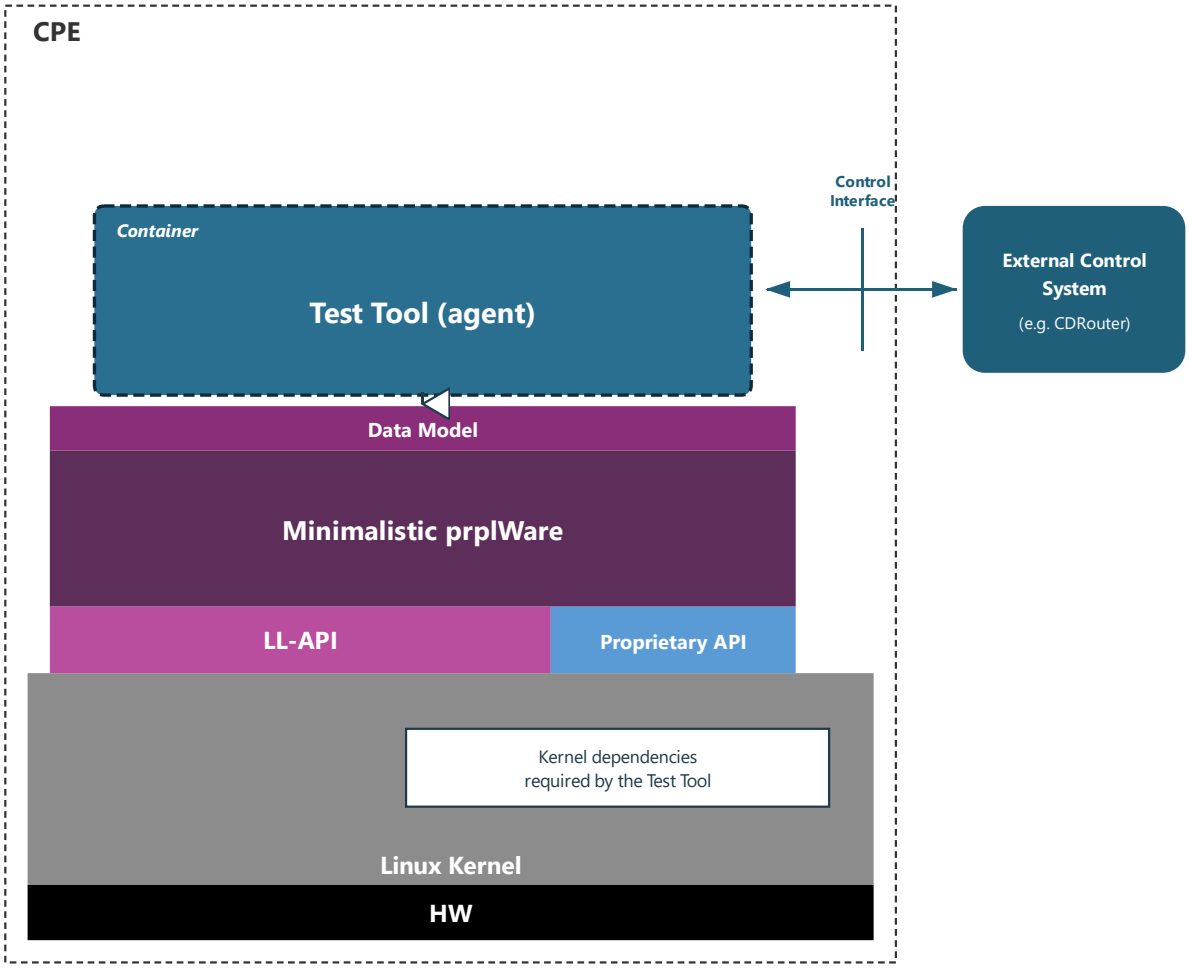
Table of Contents

Table of Contents

Results

Each test in the Low-Level API (LL-API) Test Plan must have a pass and fail metric, per Section 5.2 of the prpl Certification Program Guide. Test cases require that each test metric must pass for all the required parameters. If one parameter in the data model fails a metric in the test case, the entire test case fails.

Topology


CPE
Container
Test Tool (agent)
Control Interface
External Control System
(e.g. CDRouter)
Data Model
Minimalistic prplWare
LL-API
Proprietary API
Kernel dependencies required by the Test Tool
Linux Kernel
HW

Optional Features

This is the list of optional features and the associated test cases.

- Version Detection
  - LL-API.1.3
  - LL-API.1.4
- Factory Reset
  - LL-API.2.2
- CPU Scaling
  - LL-API.2.3

Group 1: Library Versions

Scope

The following tests identify the versions of libraries and operating systems.

Overview

The tests in this group verify the HUT (Hardware Under Test) has the correct Linux Kernel, OpenWRT, hostapd, and wpa_supplicant versions.

Test LL-API.1.1: Linux Kernel

Purpose: Verify the Linux kernel version number of the HUT.

Reference:

- https://prplfoundation.org/certification/

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Verify the Linux Kernel version.	The Linux Kernel version must adhere to the acceptable version based on the prplWare Version.

Test LL-API.1.2: OpenWRT

Purpose: Verify the OpenWRT version on the HUT.

Reference:

- https://prplfoundation.org/certification/

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Verify the OpenWRT version.	The OpenWRT version must adhere to the acceptable version based on the prplWare Version.

Test LL-API.1.3: prplMesh

Purpose: Verify the prplMesh version on the HUT.

Reference:

- https://prplfoundation.org/certification/

Functionality Tags: Version Detection

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Verify the prplMesh version.	The prplMesh version must adhere to the acceptable version based on the prplWare Version.

Test LL-API.1.4: prplLCM

Purpose: Verify the prplLCM version on the HUT.

Reference:

- https://prplfoundation.org/certification/

Functionality Tags: Version Detection

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Verify the prplLCM version.	The prplLCM version must adhere to the acceptable version based on the prplWare Version.

Test LL-API.1.5: OB-USP-A

Purpose: Verify the obuspa version on the HUT.

Reference:

- https://prplfoundation.org/certification/

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Verify the obuspa version.	The obuspa version must adhere to the acceptable version based on the prplWare Version.

Test LL-API.1.6: WPA Supplicant

Purpose: Verify the version of wpa_supplicant that must be supported.

Reference:

- prpl Low-Level API 5.7

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Verify the wireless kernel subsystem is using a valid version of wpa_supplicant.	The wpa_supplicant version must adhere to the acceptable version based on the prplWare Version.

Test LL-API.1.7: Hostapd

Purpose: Verify the version of hostapd that must be supported.

Reference:

- prpl Low-Level API 5.7

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Verify the wireless kernel subsystem is using a valid version of hostapd.	The hostapd version must adhere to the acceptable version based on the prplWare Version.

Group 2: API Verification

Scope

The following tests verify the prpl specified Low Level APIs that must be supported.

Overview

The tests in this group verify the HUT (Hardware Under Test) has properly implemented the prpl Low Level APIs.

Test LL-API.2.1: Reboot

Purpose: Verify the reboot system call on the hardware.

Reference:

- prpl Low-Level API 5.1.2

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Execute a reboot.	The HUT must reboot using the specified system calls from 5.1.2.

Test LL-API.2.2: Factory Reset

Purpose: Verify the factory reset system call on the hardware.

Reference:

- prpl Low-Level API 5.1.2

Functionality Tags: Factory Reset

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Execute a factory reset.	The HUT must perform a factory reset using the specified system calls from 5.1.2.

Test LL-API.2.3: CPU Frequency and Voltage Scaling

Purpose: Verify the CPU frequency and voltage scaling calls.

Reference:

- prpl Low-Level API 5.3.1

Functionality Tags: CPU Scaling

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Verify the support for cpu-freq subsystem by comparing time spent on sum calculation with/without boost-disable bit.	The HUT must change the cpu frequency and shorten the calculation time when boost is enabled.

Test LL-API.2.4: CPU Idle Time Management

Purpose: Verify the CPU Idle Time Management subsystem.

Reference:

- prpl Low-Level API 5.3.2

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Verify the CPU IDLE subsystem.	The HUT must perform a cpuidle using the specified system calls from 5.3.2.

Test LL-API.2.5: Wireless Capabilities

Purpose: Verify the Wireless API correctly reports hardware capabilities.

Reference:

- prpl Low-Level API 5.7

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Execute `iw phy` to see the hardware capabilities.	The HUT must follow the system calls as specified in 5.7 for reporting the hardware capabilities.

Test LL-API.2.6: Wireless Scan

Purpose: Verify the Wireless API correctly performs a Wireless scan.

Reference:

- prpl Low-Level API 5.7

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Execute a wireless scan request.	The HUT must follow the system calls as specified in 5.7 and perform a wireless scan without any errors.

Test LL-API.2.7: Ethernet Interface Management

Purpose: Verify that IP connectivity is not broken when the ethernet interface is enabled and disabled many times.

Reference:

- prpl Low-Level API 5.8.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	The ethernet interface is disabled and enabled using the netlink system calls many times.	The HUT must follow the system calls as specified in 5.8.1 for enabling and disabling the ethernet interface and confirm IPv4 connectivity.

Test LL-API.2.8: IPv4 Address Change

Purpose: Verify the IPv4 connectivity is not broken when changing the address many times.

Reference:

- prpl Low-Level API 5.8.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Change the IPv4 address using the netlink system calls.	The HUT must follow the system calls as specified in 5.8.1 for setting and then deleting the IPv4 address on an interface and confirm IPv4 connectivity.

Test LL-API.2.9: IPv4 Address Add

Purpose: Verify the IPv4 connectivity is not broken when adding and then deleting an IPv4 address many times.

Reference:

- prpl Low-Level API 5.8.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Set and delete an IPv4 address many times using the netlink system calls.	The HUT must follow the system calls as specified in 5.8.1 for setting and then deleting the IPv4 address on an interface and confirm IPv4 connectivity.

Test LL-API.2.10: IPv4 Route

Purpose: Verify that IPv4 connectivity is not broken when an IPv4 route is added and deleted many times.

Reference:

- prpl Low-Level API 5.8.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Add and delete the IPv4 routes using the netlink system calls many times.	The HUT must follow the system calls as specified in 5.8.1 for adding and deleting the IPv4 route and confirm IPv4 connectivity.

Test LL-API.2.11: IPv4 MTU

Purpose: Verify the IPv4 connectivity is not broken when changing the MTU.

Reference:

- prpl Low-Level API 5.8.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Set the IPv4 MTU using the netlink system calls.	The HUT must follow the system calls as specified in 5.8.1 for setting the IPv4 MTU and confirm IPv4 connectivity.

Test LL-API.2.12: IPv6 Address Change

Purpose: Verify the IPv6 connectivity is not broken when changing the address many times.

Reference:

- prpl Low-Level API 5.8.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Change the IPv6 address using the netlink system calls.	The HUT must follow the system calls as specified in 5.8.1 for setting and then deleting the IPv6 address on an interface and confirm IPv6 connectivity.

Test LL-API.2.13: IPv6 Address Add

Purpose: Verify the IPv6 connectivity is not broken when adding and then deleting an IPv6 address many times.

Reference:

- prpl Low-Level API 5.8.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Set and delete an IPv6 address many times using the netlink system calls.	The HUT must follow the system calls as specified in 5.8.1 for setting and then deleting the IPv6 address on an interface and confirm IPv6 connectivity.

Test LL-API.2.14: IPv6 Route

Purpose: Verify that IPv6 connectivity is not broken when an IPv6 route is added and deleted many times.

Reference:

- prpl Low-Level API 5.8.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Add and delete the IPv6 routes using the netlink system calls many times.	The HUT must follow the system calls as specified in 5.8.1 for adding and deleting the IPv6 route and confirm IPv6 connectivity.

Test LL-API.2.15: IPv6 MTU

Purpose: Verify the IPv6 connectivity is not broken when changing the MTU.

Reference:

- prpl Low-Level API 5.8.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Set the IPv6 MTU using the netlink system calls.	The HUT must follow the system calls as specified in 5.8.1 for setting the IPv6 MTU and confirm IPv6 connectivity.

Test LL-API.2.16: IPv4 Multicast Group Single Socket

Purpose: Verify that the kernel is not crashing when joining many IPv4 multicast groups on a single socket.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Join multicast groups on single socket.	The HUT must follow the system calls as specified in 5.9.1 for joining multicast groups on a single socket and not crash the kernel.

Test LL-API.2.17: IPv4 Multicast Group Multiple Sockets

Purpose: Verify that the kernel is not crashing when joining the same IPv4 multicast group on multiple sockets many times.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Join multicast groups on multiple sockets.	The HUT must follow the system calls as specified in 5.9.1 for joining multicast groups on multiple sockets and not crash the kernel.

Test LL-API.2.18: Leaving IPv4 Multicast Group

Purpose: Verify that the kernel is not crashing when joining and leaving the same IPv4 multicast groups on multiple sockets.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Join multicast groups on multiple sockets.	The HUT must follow the system calls as specified in 5.9.1 for joining multicast groups on multiple sockets and not crash the kernel.

Test LL-API.2.19: IPv4 Source Filter

Purpose: Verify that the kernel is not crashing when joining and leaving the same IPv4 multicast group with different source filters on multiple sockets.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Joining and leaving the same multicast group with different source filters on multiple sockets.	The HUT must follow the system calls as specified in 5.9.1 for joining and leaving a multicast group with different source filters on multiple sockets and not crash the kernel.

Test LL-API.2.20: IPv4 Multicast Packet Flood Single Socket

Purpose: Verify that the kernel is not crashing when joining the same IPv4 multicast group on a single socket, then receiving a large number of UDP packets on the socket.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Join the same IPv4 multicast group on a single socket, then receive a large number of UDP packets.	The HUT must follow the system calls as specified in 5.9.1 for joining a multicast group. The kernel must not crash with UDP packets sent to socket.

Test LL-API.2.21: IPv4 Multicast Packet Flood Multiple Sockets

Purpose: Verify that the kernel is not crashing when joining many IPv4 multicast groups on multiple sockets, then receiving a large number of UDP packets on the socket.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Join multiple IPv4 multicast groups on multiple sockets, then receive a large number of UDP packets.	The HUT must follow the system calls as specified in 5.9.1 for joining a multicast group. The kernel must not crash with UDP packets sent to each socket.

Test LL-API.2.22: General Query Single Socket

Purpose: Verify that the kernel is not crashing when joining an IPv4 multicast group on a single socket, then receiving a large number of General Queries.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Joining an IPv4 multicast group on a single socket, then receiving a large number of General Queries at the socket.	The HUT must follow the system calls as specified in 5.9.1 for joining a multicast group. The kernel must not crash with General Queries sent to the socket.

Test LL-API.2.23: Multicast Address Specific Query Single Socket

Purpose: Verify that the kernel is not crashing when joining an IPv4 multicast group on a single socket, then receiving a large number of Multicast Address Specific Queries.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Joining an IPv4 multicast group on a single socket, then receiving a large number of Multicast Address Specific Queries at the socket.	The HUT must follow the system calls as specified in 5.9.1 for joining a multicast group. The kernel must not crash with Multicast Address Specific Queries sent to the socket.

Test LL-API.2.24: Multicast Address and Source Specific Query Single Socket

Purpose: Verify that the kernel is not crashing when joining an IPv4 multicast group on a single socket, then receiving a large number of Multicast Address and Source Specific Queries.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Joining an IPv4 multicast group on a single socket, then receiving a large number of Multicast Address and Source Specific Queries on multiple sockets.	The HUT must follow the system calls as specified in 5.9.1 for joining a multicast group. The kernel must not crash with Multicast Address and Source Specific Queries sent to the socket.

Test LL-API.2.25: General Query Multiple Sockets

Purpose: Verify that the kernel is not crashing when joining multiple IPv4 multicast groups on multiple sockets, then receiving a large number of General Queries.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Join multiple IPv4 multicast groups on multiple sockets, then receiving a large number of General Queries at the socket.	The HUT must follow the system calls as specified in 5.9.1 for joining a multicast group. The kernel must not crash with General Queries sent to each socket.

Test LL-API.2.26: Multicast Address Specific Query Multiple Sockets

Purpose: Verify that the kernel is not crashing when joining multiple IPv4 multicast groups on a multiple sockets, then receiving a large number of Multicast Address Specific Queries.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Joining multiple IPv4 multicast groups on multiple sockets, then receiving a large number of Multicast Address Specific Queries at the sockets.	The HUT must follow the system calls as specified in 5.9.1 for joining a multicast group. The kernel must not crash with Multicast Address Specific Queries sent to each socket.

Test LL-API.2.27: Multicast Address and Source Specific Query Multiple Sockets

Purpose: Verify that the kernel is not crashing when joining multiple IPv4 multicast groups on multiple sockets, then receiving a large number of Multicast Address and Source Specific Queries.

Reference:

- prpl Low-Level API 5.9.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Joining multiple IPv4 multicast groups on multiple sockets, then receiving a large number of Multicast Address and Source Specific Queries at the sockets.	The HUT must follow the system calls as specified in 5.9.1 for joining a multicast group. The kernel must not crash with Multicast Address and Source Specific Queries sent to each socket.

Test LL-API.2.28: HMAC Template Nested

Purpose: Verify that HMAC template cannot be nested inside itself.

Reference:

- prpl Low-Level API 5.10.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Try to instantiate nested hmac algorithms using the crypto API.	The HUT must follow the system calls as specified in 5.10.1 for HW crypto API. No nested hmac algorithms were instantiated and the kernel didn't crash.

Test LL-API.2.29: Empty message with Salsa20

Purpose: Verify that an empty message can be encrypted with Salsa20 without crashing the kernel.

Reference:

- prpl Low-Level API 5.10.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Try to encrypt an empty message with Salsa20 encryption using the crypto API.	The HUT must follow the system calls as specified in 5.10.1 for HW crypto API. The empty message must be encrypted.

Test LL-API.2.30: Digest Size less than 16 bytes

Purpose: Verify that the RFC 7539 template will not be instantiated with a hash algorithm whose digest size is not 16 bytes.

Reference:

- prpl Low-Level API 5.10.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Try to instantiate an RFC7539 template with an incorrect digest size less than 16 bytes using the crypto API.	The HUT must follow the system calls as specified in 5.10.1 for HW crypto API. An RFC 7539 template must not be instantiated.

Test LL-API.2.31: VMAC transform

Purpose: Verify that a VMAC transform can be used by multiple concurrent hash requests without crashing the kernel.

Reference:

- prpl Low-Level API 5.10.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Perform multiple concurrent VMAC hash requests using the crypto API.	The HUT must follow the system calls as specified in 5.10.1 for HW crypto API. The kernel must not crash when performing the vmac hash testing.

Test LL-API.2.32: Block Size Alignment

Purpose: Verify that the kernel doesn't crash when trying to encrypt a message with a size not aligned to the block cipher's block size, and where the destination buffer starts exactly at a page boundary.

Reference:

- prpl Low-Level API 5.10.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Encrypt a message with a size that doesn't align to the block cipher's block size to a destination buffer that starts at a page boundary using the crypto API.	The HUT must follow the system calls as specified in 5.10.1 for HW crypto API. The kernel must not crash when performing the encryption.

Test LL-API.2.33: Malformed Authentication Keys

Purpose: Verify the kernel does not crash when setting malformed authentication keys.

Reference:

- prpl Low-Level API 5.10.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Set malformed authentication keys using the crypto API.	The HUT must follow the system calls as specified in 5.10.1 for HW crypto API. The kernel must not crash during the settings and the malformed keys calls must not be successful.

Test LL-API.2.34: Uninitialized Memory

Purpose: Verify the crypto library does not leak uninitialized memory into user space.

Reference:

- prpl Low-Level API 5.10.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Use NLM_F_DUMP mode to get all algorithms and check all the string fields for unexpected nonzero bytes.	The HUT must follow the system calls as specified in 5.10.1 for HW crypto API. No memory leaks are detected.

Test LL-API.2.35: Null Pointer Dereference

Purpose: Verify the crypto library does not have a null pointer dereference in the kernel.

Reference:

- prpl Low-Level API 5.10.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Use the CRYPTO_MSG_DELALG message in the NETLINK_CRYPTO interface to try to delete a test algorithm.	The HUT must follow the system calls as specified in 5.10.1 for HW crypto API. The kernel must not crash.

Test LL-API.2.36: Storage Volumes

Purpose: Verify that the storage API correctly identifies the flash chip and the partition table defined in the Device Tree (DTS).

Reference:

- prpl Low-Level API 5.11.1

Functionality Tags: Core

Test Setup: Common Test Setup is used at the beginning of each part.

Procedure:

Step	Action	Expected Behavior
1.	Execute mtd commands to see the partition layout.	The HUT must follow the system calls specified in 5.11. The listed partition names and sizes must match the prplOS partition layout specification.

References

- prpl Low-Level API v2
  - https://prplfoundation.org/wp-content/uploads/prpl-Low-Level-API-Specification-v2.pdf