



Apertis Platform Technical Vision

1	Contents	
2	Overview	2
3	Fixed function device scenario	2
4	HMI device scenario	4
5	Focus area Flatpak applications	5
6	(Industrial) IOT scenario	6
7	Focus area container runtime	6
8	SDK scenario	7

9 Overview

10 The intention of this document is to outline the Apertis technical direction and
 11 vision for the Apertis platform as it would be used on a device, container or
 12 VM.

13 This document does not cover the overall Apertis infrastructure and its general
 14 principles around for example [open source license expectations](#)¹, [image build-](#)
 15 [ing](#)² and the [release process](#)³. These topics are covered in their own respective
 16 documents.

17 Apertis is, by design, a very flexible platform to build on. However, as a project
 18 we cannot directly support and test every possible setup that can be built using
 19 Apertis. Because of that our development and testing is focused on a few specific
 20 [hardware reference platforms](#)⁴ which can be used as the basis to explore a wide
 21 range of use-cases. Apertis users are of course still free to choose their hardware
 22 platforms and configure their systems differently, utilising as much or as little
 23 of that provided by Apertis as they see fit!

24 The following sections look at various scenarios in which Apertis can be used on
 25 devices and the technical building blocks that Apertis intends to make available
 26 for them. As such they all describe a reasonably full-featured setup for each
 27 scenario, however an actual deployment can still choose to only use a subset of
 28 available features.

29 As this document provides a forward-looking vision of the Apertis platform not
 30 all features mentioned are yet implemented or even fully defined.

¹<https://martyn.pages.apertis.org/apertis-website/policies/license-expectations/>

²https://martyn.pages.apertis.org/apertis-website/guides/image_building/

³https://martyn.pages.apertis.org/apertis-website/guides/apertis_release_process/

⁴https://martyn.pages.apertis.org/apertis-website/reference_hardware/

31 Fixed function device scenario

32 This setup represents an “appliance” or fixed function devices. That is to say
33 the device is intended to be used for a specific functionality which cannot be ex-
34 tended by installing extra software on top of the base system. This scenario also
35 assumes it’s operating in a headless fashion without a comprehensive graphical
36 user interface. It is still possible to have some amount of user interactions for ex-
37 ample via simple buttons or knobs for user inputs and lights and/or segmented
38 displays for outputs, however typical inputs and outputs for these devices will
39 be attached peripherals and sensors.

40 Typically these devices are expected to be connected to an IP network and have
41 the ability to either directly or indirectly connected to internet services (e.g. via
42 ethernet, wifi, 5G).

43 As security and integrity is paramount the device supports a fully verified boot
44 sequence (secure boot, verified boot or similar) to ensure untampered firmware,
45 kernel, etc are used. The system (root) filesystem is integrity measured for all
46 executables to get a fully trusted system.

47 To further improve integrity and privacy of the system a Trusted Execution envi-
48 ronment is available for trusted applications (e.g. optee). The TEE environment
49 or a dedicated chip (e.g. TPM) are available to support remote attestation as
50 well support for encrypted areas (filesystems etc) which can only be accessed by
51 a specific device and only when it has been booted into a known state.

52 On the running OS [systemd](https://www.freedesktop.org/wiki/Software/systemd/)⁵ for overall system startup and management. This
53 also includes managing the usage of linux capabilities, resource constraints, sys-
54 tem call filtering, sandboxing of services through linux namespaces and provid-
55 ing start-on-demand as well as watchdog services.

56 On top of systemd the Apparmor LSM is used to further constraint the be-
57 haviour of system processes and provide a second line of protection increasing
58 the defense in depth.

59 In the current world of software nothing is secure if it’s not also up to date
60 with the latest security fixes. As such the system comes with an ostree based
61 over the air update system with the capability to integrate with cloud device
62 management and fleet management systems such as Hawkbit.

63 Building upon the fleet management integration, while updates provide one
64 piece of the puzzle telemetric support provides another aspect to enable remote
65 management and monitoring of devices.

66 As a lot of this functionality needs network access, so of course Apertis supports
67 various ways of accessing networks whether this is via wired connection, wifi or
68 mobile networks.

⁵<https://www.freedesktop.org/wiki/Software/systemd/>

feature	documentation	status
Verified boot sequence	fully verified boot sequence ⁶	Implemented
Integrity validate root filesystem	Security ⁷	Concept: Requires Update
Filesystem Encryption		
Trusted Execution Environment	Trusted Execution Environment ⁸	Concept: Up-to-date
System service lifecycle management	system startup and management ⁹	Partially Implemented
Apparmor (LSM)	Apparmor ¹⁰	Implemented
OTA/system upgrade	OSTree based ¹¹	Implemented
OTA Fleet management	Preparing hawkBit for Production Use ¹²	Concept: Up-to-date
Network connectivity	Connectivity ¹³	Concept: Requires Update

69 HMI device scenario

70 This setup is targeted at devices with a HMI. Typically this will be a modern
71 graphical user interface driven via a touch-screen and/or other inputs as well as
72 the capability of audio in and outputs. Some devices may also have one or more
73 cameras or other sensors attached. Furthermore these devices can be extended
74 via the installation of user-facing applications.

75 As a basis these devices have all the features and capabilities of the fixed function
76 device scenario.

77 On top of this base functionality a user interface is available to launch different
78 applications or functions via a touch-screen or other inputs. This user interface is
79 composed from a base wayland compositor which can be re-used and customised
80 for specific projects, for Apertis a reference UX shell is available in the form of
81 the maynard compositor.

82 For video inputs as well as audio input and output pipewire is used as a routing
83 daemon with wireplumber adding policy support. This allows multiple applica-
84 tions to use these streams at the same time while also being able to prioritise
85 between them and implement more complex policies like for example audio
86 ducking.

87 The additions of applications to the system can be done via the installation of
88 flatpak-based application bundles, which allows applications to be installed with
89 minimal dependencies on the base system. This also makes it possible to have

⁶<https://martyn.pages.apertis.org/apertis-website/architecture/secure-boot/>

⁷<https://martyn.pages.apertis.org/apertis-website/concepts/security/>

⁸<https://martyn.pages.apertis.org/apertis-website/concepts/op-tee/>

⁹https://martyn.pages.apertis.org/apertis-website/architecture/boot_process/

¹⁰<https://martyn.pages.apertis.org/apertis-website/guides/apparmor/>

¹¹<https://martyn.pages.apertis.org/apertis-website/guides/ostree/>

¹²<https://martyn.pages.apertis.org/apertis-website/concepts/preparing-hawkbit-for-production/>

¹³<https://martyn.pages.apertis.org/apertis-website/concepts/connectivity/>

90 separate update lifecycles for applications and the base operating system as typ-
 91 ically applications are updated at a far shorter cycle then the operating system
 92 itself. The usage of flatpak-based applications also enables the implementation
 93 of dynamic policies for what resources are available for an application. For
 94 example camera access might only be allowed for some applications.

95 Flatpak applications can either be provisioned via a fleet management system
 96 (such as Hawkbit) or from a “app store” application available on the system
 97 with application specific download methods.

feature	documentation	status
Reference HMI shell and compositor	Application Framework ¹⁴	Concept: Up-to-date
Bluetooth	Connectivity: Bluetooth Support ¹⁵	Concept: Requires Update
System toolkit		
Virtual system keyboard		
Audio routing and policy	Audio management ¹⁶	Concept: Up-to-date
Video routing and policy		
Application framework integration	Application framework ¹⁷	Concept: Up-to-date
Application management (store or fleet)	hawkBit ¹⁸	Partially Implemented
Removable storage management		

98 **Focus area Flatpak applications**

99 As Flatpak applications are decoupled from the base system they are essentially
 100 their own dedicated setup.

101 To be able to easily build Flatpak applications targeting Apertis systems, while
 102 still taking the benefits of the Apertis maintenance, dedicated Apertis runtime
 103 including SDK variants and debug extensions (needed for development) will be
 104 provided. These runtime will include a basic set of libraries for libraries to rely
 105 on.

106 However certain applications or devices can have special needs for the libraries
 107 available in their standard runtime. Custom runtimes can also be created as
 108 needed.

109 For some devices specific extra or different libraries can be required. For example
 110 to support a device specific GL or Vulkan stack or device specific codec libraries.
 111 These can be shipped as a flatpak runtime extension, allowing multiple devices

¹⁴<https://martyn.pages.apertis.org/apertis-website/concepts/application-framework/#compositor-libweston>

¹⁵<https://martyn.pages.apertis.org/apertis-website/concepts/connectivity/#bluetooth-support>

¹⁶<https://martyn.pages.apertis.org/apertis-website/concepts/audio-management/>

¹⁷<https://martyn.pages.apertis.org/apertis-website/concepts/application-framework/>

¹⁸<https://martyn.pages.apertis.org/apertis-website/guides/deployment-management/>

112 to use the same base runtimes but adjust as needed to take full advantage of
113 the underlying hardware.

114 Apart from this base infrastructure application may also have a need to access
115 peripherals, sensors, audio inputs and outputs as well as other lower-level system
116 interfaces. For this purpose flatpaks concepts of portals will be used, which
117 allows the system to apply applications specific policies and permissions. An
118 example policy is whether a given application can access the devices camera
119 potentially only after explicit user interaction.

feature	documentation link	status
Apertis supported Flatpak runtimes		
Runtime extensions for HW specific support		
Flatpak System API access (portals)		
Flatpak audio and video routing		
Creation of Flatpaks		

120 **(Industrial) IOT scenario**

121 Another area in which Apertis focuses is industrial IOT. In a sense these devices
122 are close to fixed function devices in that they've got little to no ability to
123 support user interaction. However these are not fixed function devices, these
124 devices are targeted at running "edge" workloads with the ability to dynamically
125 manage which devices run specific workloads.

126 Typically these devices collect data on the edge either directly by containing var-
127 ious sensors (e.g. cameras, power measurement, temperature etc) or indirectly
128 via other devices on a local network (which could be fixed function apertis de-
129 vices). The role of these devices typically is to process all these inputs in some
130 fashion and either act on them or relay it to a cloud infrastructure.

131 To allow workloads to be flexible they will be executed as containerized work-
132 loads; These containers are distributed and run using the open container initia-
133 tive (OCI) standards. Apertis will include an open-source workload orchestrator
134 to interact with a management system to orchestrate deployments.

135 Another role an edge device can take is as an orchestrator for deploying soft-
136 ware updates or workloads to secondary devices. For example an accompanying
137 microcontroller running a real time operating system or a separate network-
138 attached system which is not (or cannot be) directly connected to the internet.
139 The orchestrator is able to manage and push updates to these devices.

feature	documentation link	status
OCI container orchestration		
Remote firmware deployment		

feature	documentation link	status
Remote workload deployment		
OCI container management		

140 Focus area container runtime

141 Like Flatpaks OCI images to be run on a device are separate from the main
 142 system to decouple the two. OCI images are the standard way of distributing
 143 cloud workload for network services which is a good and natural fit for the
 144 workloads on edge devices. This allows the Apertis device to take advantage of
 145 common workflows developers are used to for building images targeting cloud
 146 deployments.

147 The other benefit of using OCI standards is the potential to use/consider dif-
 148 ferent [CRI-O](#)¹⁹ implementation, including implementations using hypervisor
 149 technology to allow for an even bigger separation of host and container then
 150 typically available on namespace based container systems.

feature	documentation link	status
Guidelines for OCI container deployments		
OCI container building		
OCI container integration with system API		

151 SDK scenario

152 The final area is the SDK environment; While not really a product as such,
 153 ease of development is a critical aspect for the success of Apertis. To enable
 154 smooth development Apertis provides a pre-build virtual machine for develop-
 155 ment (which in a sense is yet another device). This is preinstalled with all the
 156 tooling required to locally build all aspects of the Apertis universe such as:

- 157 • Build Debian packages for target devices
- 158 • Build full system images
- 159 • Build Flatpak applications
- 160 • Build OCI images

161 Apart from building, deployment and debugging should be as easy as possible.
 162 To support that, tooling is included to directly provision the various artifacts
 163 to local devices (depending on the device installation of course).

feature	documentation link	status
Debian package building (devroot, sysroot)	sysroots and devroots ²⁰	Implemented

¹⁹<https://cri-o.io/>

feature	documentation link	status
System image building (debos)	Image building ²¹	Implemented
Flatpak image building		
OCI image building		
SDK development environment (IDE)	SDK ²²	Implemented
Local Device deployment/debugging	ADE ²³	Implemented: Requires Update

²⁰<https://martyn.pages.apertis.org/apertis-website/architecture/sysroots-and-devroots/>

²¹https://martyn.pages.apertis.org/apertis-website/guides/image_building/

²²<https://martyn.pages.apertis.org/apertis-website/guides/virtualbox/>

²³<https://martyn.pages.apertis.org/apertis-website/guides/ade/>