



HWpack Requirements

# 1 Contents

2	<b>Concepts</b>	<b>2</b>
3	Image Building Stages . . . . .	2
4	OSpack . . . . .	2
5	HWpack . . . . .	3
6	Apertis packages . . . . .	3
7	Typical Apertis OS layout . . . . .	4
8	<b>HWpack Components</b>	<b>5</b>
9	Bootloader . . . . .	5
10	Linux Kernel . . . . .	5
11	Firmware . . . . .	6
12	Debos .yaml configuration . . . . .	6
13	Documentation . . . . .	7
14	<b>Testing</b>	<b>7</b>
15	<b>Licensing</b>	<b>7</b>

16 This documentation covers the requirements and considerations that should be  
17 taken into account when implementing “hardware packs” for the Apertis project.

## 18 Concepts

19 This section briefly covers the concepts and expectations of the Apertis platform.

### 20 Image Building Stages

21 Apertis images are built from binary packages, packaged in the `.deb` format.  
22 Building these packages is expected to be carried out from source by the Apertis  
23 infrastructure, ensuring all packages dependencies are properly described and  
24 reducing the risk of unexpected dependencies.

25 The selection and packaging of these packages are predominantly driven by the  
26 needs of the two main process steps required to create images, known as the  
27 `OSpack` and `HWpack`.

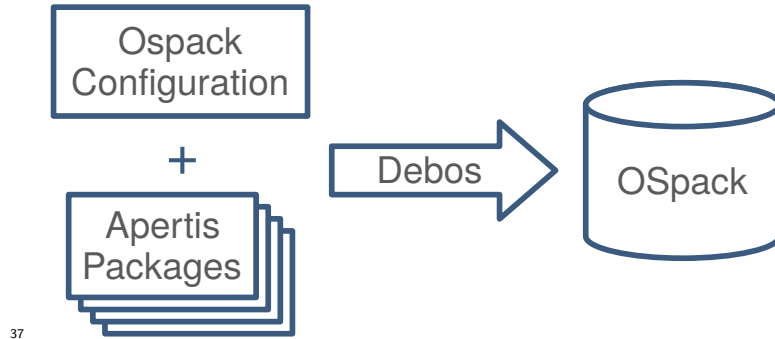
### 28 OSpack

29 The `OSpack` stage generates one or more generic (architecture specific but  
30 largely hardware independent) archived rootfs built from Apertis packages.  
31 These rootfs archives are known as `OSpacks`. The process is managed by a  
32 tool called `Debos`<sup>1</sup>, which uses `yaml` configuration files to guide what steps  
33 it takes. Apertis provides `yaml` files to assemble a number of differently tar-  
34 getted `OSpacks`, ranging from a minimal GUI-less `OSpack`, a target focused GUI

---

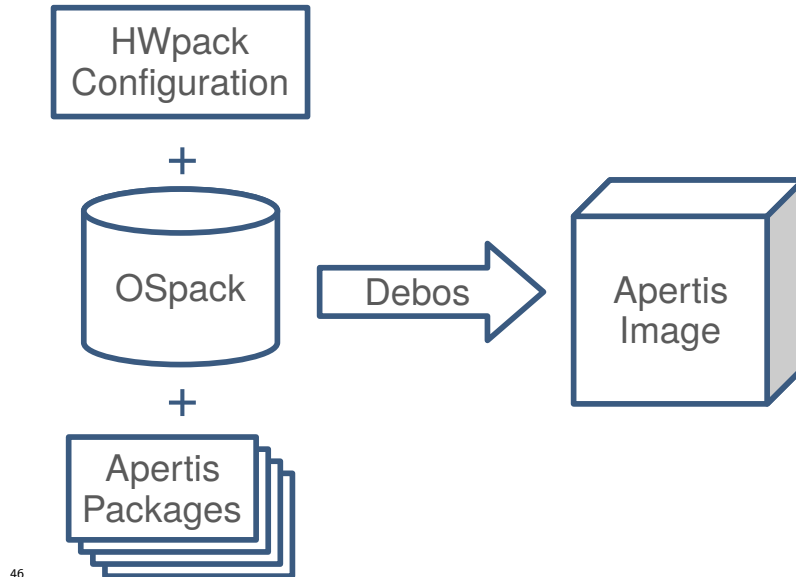
<sup>1</sup><https://github.com/go-debos/debos>

35 OSpack and a development environment with a desktop style GUI and has pre-  
36 packaged the components required to generate these OSpacks.



### 38 HWpack

39 Unlike the OSpack step, the hardware package (HWpack) step does not result  
40 in an item known as a HWpack. The HWpack is comprised of a Debos script  
41 which controls the processing of a run time determined OSpack to convert it  
42 from a hardware independent OSpack into an image which can be successfully  
43 booted on a specific hardware platform. In addition to developing the HWpack  
44 script, the HWpack step requires the modification and packaging of the required  
45 components to perform this transformation.



## 47 **Apertis packages**

48 Apertis standardizes on a specific set of components (with specific versions) and  
49 technologies to fulfill the needs of the target platforms. This maximizes sharing  
50 and reuse, thus minimizing effort and cost of maintaining common components  
51 across products. Deviations from the standard selection may be needed to ac-  
52 commodate product-specific needs but such deviations tend to reduce reuse and  
53 thus increase the long-term maintenance efforts. It will likely fall on the product  
54 team in question to carry this added effort. As a result, it is strongly preferred  
55 for deviations to be minimized with generic improvements and additions made  
56 to the standard components to add the required functionality where possible.

57 The components selected as part of the base Apertis system need to meet a  
58 number of project criteria.

59 **Licensing requirements** Components need to be licensed in such a way that  
60 they are acceptable for distribution in target devices. For example, GPL-3  
61 is problematic and thus avoided.

62 **Software revisions** The specific revisions of the packages are picked to bal-  
63 ance the competing customer needs of having up-to-date versions (and  
64 thus features), stability and the need for a strong security road map.

65 **Close to upstream** Apertis aims to remain relatively close to its upstreams  
66 (where the majority of packages are based on Debian stable, the kernel  
67 on the latest LTS release). This minimizes the effort required to migrate  
68 to newer versions as it means there are minimal patches to port. A large  
69 deviation from upstream also decreases effectiveness of testing and the  
70 validity of review performed on upstream versions.

71 These are some of the key packages from which the Apertis system is built:

- 72 • U-Boot/systemd-boot
- 73 • Linux kernel
- 74 • systemd
- 75 • Apparmor
- 76 • Wayland
- 77 • Mesa
- 78 • PulseAudio

79 All Apertis packages are packaged using standard Debian packaging, with source  
80 code and package configuration stored in the Apertis GitLab enabling automa-  
81 tion of the package build process.

## 82 **Typical Apertis OS layout**

83 The reference Apertis images share a common layout per architecture, enabling  
84 images to be shared across the various supported platforms of each architecture:

- 85 • Bootloader typically stored in flash

- 86 • Kernel and other boot components and configuration stored on rootfs  
87 (enabling current update mechanism)
- 88 • OSTree used as part of update strategy and rollback (non-OSTree options  
89 available for development)

90 It is expected that the requirements and practicalities of products based on  
91 Apertis will require deviations to be made from this layout. Such deviations  
92 however should be carefully considered. Some, such as storing the bootloader  
93 at the beginning of the same medium as the rootfs, carry very little impact as  
94 far as the functionality of Apertis is concerned. Others such as using a different  
95 bootloader, storing the kernel outside of the rootfs or using a different update  
96 strategy (such as A/B partitioning) may pose in a non-trivial effort for inte-  
97 gration, loss of some Apertis functionality and/or extra on-going maintenance  
98 effort.

99 The OSpack is expected to contain common functionality to enable use of sup-  
100 ported hardware, for example the OSpacks which are intended to be used with  
101 an operational graphical environment include Wayland, though the hardware  
102 specific drivers are in the HWpacks. When enabling new types of functional-  
103 ity, it is expected that generic support would be added to the OSpacks where  
104 applicable. If such functionality is widely used, this should be integrated into  
105 the Apertis OSpacks. Support for niche functionality, or functionality not of  
106 general interest to Apertis, will need to be added to a product specific OSpack.

## 107 HWpack Components

108 As with the OSpack (and unless specific exceptions provided) all components  
109 should be properly packaged and provided with source to enable debugging,  
110 extension and further optimization. It is expected that some changes may be  
111 viable to be included in the main Apertis packages, some packages may be added  
112 to the main Apertis package repositories and others will need bespoke packages  
113 which would typically be stored in a dedicated project area, as described in  
114 the [contribution process](#)<sup>2</sup> document. It is typical for the following areas to  
115 need modifications or to be provided, though other modifications may also be  
116 required.

### 117 Bootloader

118 Apertis standardizes on the U-Boot as the bootloader on all non-x86 platforms.  
119 In order to support the standard Apertis boot, update and rollback functionality  
120 it is necessary for the configuration to include the “[Generic Distro Configuration  
121 Concept](#)”<sup>3</sup> (often referred to as “Distro Boot”). The configuration used by this  
122 mechanism has been tweaked to work with Apertis rollback mechanisms.

---

<sup>2</sup><https://jwd.pages.apertis.org/apertis-website/policies/contributions/>

<sup>3</sup><https://gitlab.apertis.org/pkg/u-boot/blob/apertis/v2020pre/doc/README.distro>

123 In order to enable efficient development, it would be advisable to ensure that  
124 access to the boot prompt is enabled along with networking support and the  
125 PXE and DHCP boot options where applicable. (Note: U-Boot can supports  
126 networking via a USB RNDIS gadget should a USB On-The-Go (USB OTG)  
127 port be available.)

## 128 **Linux Kernel**

129 Apertis expects projects that using it have a need to take product security  
130 seriously, as a result known kernel vulnerabilities need to be patched and updates  
131 made available. Apertis uses and tracks the latest upstream [longterm](#)<sup>4</sup> stable  
132 (LTS) kernel available at time of a release being made. The Apertis project  
133 strongly recommends that when products use their own kernel, these are kept  
134 as close to the upstream kernel as possible and preferably based on an LTS  
135 kernel.

136 It is understood that in some circumstances it may be necessary to utilize a  
137 heavily modified “vendor kernel”. Please note that these kernels are typically  
138 not provided with any form of long-term support and thus may quickly lack  
139 important security and stability fixes. Unless otherwise agreed, the burden of  
140 supporting such kernels will remain with the product team. Likewise, in addition  
141 to lacking a source of security fixes, many older kernels are known to have serious  
142 vulnerabilities that can only be fully resolved/mitigated by updating to a newer  
143 kernel. Apertis strongly discourages the use of such kernels.

144 The Apertis kernel contains a number of modifications primarily to enhance the  
145 Apparmor support provided by the upstream kernel. The patches used by the  
146 stock Apertis kernel can be found in the Apertis [GitLab](#)<sup>5</sup>. In order to support  
147 Apertis’ use of Apparmor, a kernel needs to support the following Apparmor  
148 mediations:

- 149 • file
- 150 • ptrace
- 151 • signal
- 152 • dbus
- 153 • network
- 154 • capability
- 155 • mount
- 156 • umount
- 157 • namespaces

158 Additionally, the kernel should be configured to support the [functionality re-](#)  
159 [quired by systemd](#)<sup>6</sup>.

---

<sup>4</sup><https://www.kernel.org/category/releases.html>

<sup>5</sup><https://gitlab.apertis.org/pkg/linux/tree/apertis/v2020pre/debian/patches/apparmor>

<sup>6</sup><https://gitlab.apertis.org/pkg/systemd/blob/apertis/v2020pre/README>

160 For development purposes, the kernel should provide early serial debugging and  
161 be capable of booting from an NFS rootfs.

## 162 **Firmware**

163 It is understood that many hardware platforms may need firmware, provided  
164 by the vendor as binaries, to use certain functionality provided by the device.  
165 It is still expected that such firmware is packaged as a deb package, though it is  
166 understood that source will not be available for such components. The Apertis  
167 infrastructure should still be used to build the binary packages.

## 168 **Debos .yaml configuration**

169 Apertis uses Debos to automate the conversion of binary packages into images  
170 suitable for installation on specific targets in several stages. The configuration  
171 used for the Apertis reference platforms can be found in [GitLab](#)<sup>7</sup> with their use  
172 documented in `README.md`. It is expected that a HWpack provides configuration  
173 file(s) that:

- 174 • Generate the required image(s) from either a reference or project specific  
175 OSpack
- 176 • Generate images containing the partitioning expected by the target and  
177 project
- 178 • Add any extra components needed via the installation of packages
- 179 • Are provided with any scripts required to aid in the application of minor  
180 changes to tweak the image to required default configuration
- 181 • Generate any project specific OSpacks when sufficient support can't be  
182 added to the generic OSpack recipes to cover the functionality required  
183 by the relevant project.

## 184 **Documentation**

185 Documentation should be provided with the Debos configuration detailing the  
186 use of any configuration files provided and documenting the process to be fol-  
187 lowed to install the generated images into a new target device to yield a booting  
188 system.

## 189 **Testing**

190 Apertis provides infrastructure to both continuously build and test software  
191 on target devices based on [Docker](#)<sup>8</sup>, [GitLab CI/CD](#)<sup>9</sup> and [LAVA](#)<sup>10</sup>. It is ex-  
192 pected that the provided source and configuration artifacts (and possibly bi-

---

<sup>7</sup><https://gitlab.apertis.org/infrastructure/apertis-image-recipes>

<sup>8</sup><https://www.docker.com/>

<sup>9</sup><https://docs.gitlab.com/ee/ci/>

<sup>10</sup><https://lavasoftware.org/>

193 nary firmware as mentioned above), when integrated into the provided Apertis  
194 infrastructure, will be capable of generating images which pass hardware boot  
195 testing with no manual steps required.

## 196 **Licensing**

197 Code, including build scripts, helpers and recipes, developed for Apertis should  
198 comply with the [Apertis Licensing](#)<sup>11</sup>.

---

<sup>11</sup><https://jwd.pages.apertis.org/apertis-website/policies/license-applying/>