



Preparing hawkBit for Production Use

# 1 Contents

2	<b>Introduction</b>	<b>2</b>
3	<b>Evaluation Report</b>	<b>2</b>
4	Server configuration . . . . .	2
5	Considering the production workflow . . . . .	3
6	Management UI access . . . . .	4
7	Enabling device filtering . . . . .	4
8	Provisioning for multiple product teams or partners . . . . .	5
9	Life management of artifacts . . . . .	5
10	Platform scalability . . . . .	5
11	<b>Recommendation</b>	<b>6</b>
12	Server Configuration . . . . .	6
13	Considering the production workflow . . . . .	6
14	Management UI access . . . . .	6
15	Enabling device filtering . . . . .	6
16	Provisioning for multiple product teams or partners . . . . .	6
17	Life management of artifacts . . . . .	7
18	Platform scalability . . . . .	7

## 19 Introduction

20 The Apertis project has been experimenting with the use of [Eclipse hawkBit](https://www.eclipse.org/hawkbit/)<sup>1</sup> as  
21 a mechanism for the deployment of system updates and applications to target  
22 devices in the field. The current emphasis is being placed on system updates,  
23 though hawkBit can also be used to address different software distribution use  
24 cases such as to distribute system software, updates and even apps from an app  
25 store.

26 Apertis has recently deployed a [hawkBit instance](https://hawkbit.apertis.org)<sup>2</sup> into which the [image build  
27 pipelines](https://gitlab.apertis.org/infrastructure/apertis-image-recipes/-/pipelines)<sup>3</sup> are uploading builds. The [apertis-hawkBit-agent](https://gitlab.apertis.org/pkg/apertis-hawkbit-agent)<sup>4</sup> has been added to  
28 OSTree based images and a guide produced detailing how this can be used to  
29 [deploy updates to an Apertis target](https://sjoerd.pages.apertis.org/apertis-website/guides/deployment-management/)<sup>5</sup>.

30 The current instance is proving valuable for gaining insight into how hawkBit  
31 can be used as part of the broader Apertis project. hawkBit is already in use  
32 elsewhere, notably by [Bosch as part of its IoT infrastructure](https://docs.bosch-iot-rollouts.com/documentation/index.html)<sup>6</sup>, however more  
33 work is required to reach the point where the Apertis infrastructure (or a deploy-  
34 ment based on the Apertis infrastructure) would be ready for production use. In

---

<sup>1</sup><https://www.eclipse.org/hawkbit/>

<sup>2</sup><https://hawkbit.apertis.org>

<sup>3</sup><https://gitlab.apertis.org/infrastructure/apertis-image-recipes/-/pipelines>

<sup>4</sup><https://gitlab.apertis.org/pkg/apertis-hawkbit-agent>

<sup>5</sup><https://sjoerd.pages.apertis.org/apertis-website/guides/deployment-management/>

<sup>6</sup><https://docs.bosch-iot-rollouts.com/documentation/index.html>

35 this document we will describe the steps we feel that need to be taken to provide  
36 a reference deployment that could be more readily suitable for production.

## 37 Evaluation Report

### 38 Server configuration

39 The current hawkBit deployment is hosted on Collabora's infrastructure. The  
40 example [Docker Compose configuration file](#)<sup>7</sup> has been modified to improve sta-  
41 bility, security and adding a reverse proxy providing SSL encryption. This has  
42 been wrapped with [Chef](#)<sup>8</sup> configuration to improve maintainability. Whilst this  
43 configuration has limitations (that will be discussed later), it provides a better  
44 starting point for the deployment of a production system. These configuration  
45 files are currently stored in Collabora's private infrastructure repository and  
46 thus not visible to 3rd parties.

### 47 Considering the production workflow

48 The currently enabled process for the enrollment and configuration of a target  
49 device into the hawkBit deployment infrastructure requires the following steps:

- 50 • Install Apertis OSTree based image on the target device.
- 51 • Define or determine the `controllerid` for the device. This ID needs to be  
52 unique on the hawkBit instance as it is used to identify the target.
- 53 • Enroll the target on the hawkBit instance, either via the [UI](#)<sup>9</sup> or [API](#)<sup>10</sup>.  
54 – If adding via the UI, hawkBit creates a security token, if adding via  
55 the API the security token can be generated outside of hawkBit.
- 56 • Modify the configuration file for `apertis-hawkbit-agent` to contain the cor-  
57 rect URL for the hawkBit instance, the targets `controllerid` and the  
58 generated security token. This configuration file is `/etc/apertis-hawkbit-  
59 agent.ini`. Without these options being set, the target will be unable to  
60 find and access the deployment server to discover updates.

61 This workflow presents a number of points that could prove contentious in a  
62 production environment:

- 63 • A need for access to the hawkBit deployment server (that may be hosted on  
64 external cloud infrastructure) from the production environment to register  
65 the `controllerid` and security token.
- 66 • The requirement to have a mechanism to add configuration to the device  
67 post software load.

---

<sup>7</sup>[https://github.com/eclipse/hawkbit/blob/master/hawkbit-runtime/docker/docker-  
compose-stack.yml](https://github.com/eclipse/hawkbit/blob/master/hawkbit-runtime/docker/docker-compose-stack.yml)

<sup>8</sup><https://www.chef.io/>

<sup>9</sup><https://www.eclipse.org/hawkbit/ui/#deployment-management>

<sup>10</sup>[https://www.eclipse.org/hawkbit/rest-api/targets-api-guide/#\\_post\\_rest\\_v1\\_targets](https://www.eclipse.org/hawkbit/rest-api/targets-api-guide/#_post_rest_v1_targets)

68 The security token based mechanism is one of a [number of options](#)<sup>11</sup> available  
69 for authentication via the DDI API. The security token must be shared between  
70 the target and the hawkBit server. This approach has a number of downsides:

- 71 • The Token needs to be added to the hawkBit server and tied to the target  
72 devices `controllerid`. This may necessitate a link between the production  
73 environment and an external network to access the hawkBit server.
- 74 • The need for the shared token to be registered with the server for authen-  
75 tication would make it impossible to use the “plug n’ play” enrollment of  
76 the target devices supported by hawkBit.

77 hawkBit allows for a certificate based authentication mechanism (using a re-  
78 verse proxy before the hawkBit server to perform authentication) which would  
79 remove the need to share a security token with the server. Utilizing signed keys  
80 would allow authentication to be achieved independently from enrollment, thus  
81 allowing enrollment to be carried out at a later date and would remove the need  
82 to store data per device in the hawkBit from the production environment. hawk-  
83 Bit allows for “[plug’n play](#)”<sup>12</sup> enrollment, the enrollment of the device when it’s  
84 first seen by hawkBit, thus the device could potentially be enrolled once the end  
85 user has switched on the device and successfully connected it to a network for  
86 the first time when using certificate based authentication.

87 For many devices it would not be practical or desired to have remote access  
88 into the production firmware to add device specific configuration, such as a  
89 security token or device specific signed key. `apertis-hawkbit-agent` currently  
90 expects such configuration to be saved in `/etc/apertis-hawkbit-agent.ini`. An  
91 option that this presents is for the image programmed onto the target to provide  
92 2 OSTree commits, one with the software expected on the device when shipped  
93 and the other for factory use, with boot defaulting to the latter. OSTree will  
94 attempt to merge any local changes made to the configuration when updating  
95 the image. The factory image could be used to perform any testing and factory  
96 configuration tasks required before switching the device to the shipping software  
97 load. Customizations to the configuration made in the factory should then be  
98 merged as part of the switch to the shipping load, and the factory commit can be  
99 removed from the device. Such an approach could provide some remote access  
100 to the target as part of the factory commit, but not the shipping commit, thus  
101 avoiding remote access being present in the field.

102 As previously mentioned, a unique `controllerid` is needed by hawkBit to identify  
103 the device and needs to be stored in the configuration file. An alternative  
104 approach may be to generate this ID from other unique data provided by the  
105 device, such as a MAC address or unique ID provided by the SoC used in the  
106 device.

---

<sup>11</sup><https://www.eclipse.org/hawkbit/concepts/authentication/>

<sup>12</sup><https://github.com/eclipse/hawkbit/archives/2016/07/27>

## 107 **Management UI access**

108 We currently have a number of static users defined with passwords available to  
109 trusted maintainers. Such as scheme is not going to scale in a production envi-  
110 ronment, nor provide an adequate level of security for a production deployment.  
111 hawkBit provides the ability to configure authentication using a provider imple-  
112 menting the OpenID Connect standard, which would allow for much greater  
113 flexibility in authenticating users.

## 114 **Enabling device filtering**

115 hawkBit provides functionality to perform update rollouts in a controlled way,  
116 allowing a subset of the deployed base to get an update and only moving on to  
117 more devices when a target percentage of devices have received the update and  
118 with a configurable error rate. When rolling out updates, in an environment  
119 where more than one hardware platform or revision of hardware is present, it  
120 will be necessary to be able to ensure the correct updates are targeted towards  
121 the correct devices. For example, two revisions of a gadget could use different  
122 SoCs with different architectures each requiring a different build of the update  
123 and different versions of a device may need to be updated with different streams  
124 of updates. In order to cater for such scenarios, it is important for hawkBit to be  
125 able to accurately distinguish between differing hardware. Support to achieve  
126 this is provided via hawkBit's ability to store attributes. These attributes can be  
127 set by the target device via the DDI interface once enrolled and used by hawkBit  
128 to filter target devices into groups. At the moment the `apertis-hawkbit-agent` is  
129 not setting any attributes.

## 130 **Provisioning for multiple product teams or partners**

131 In order to use hawkBit for multiple products or partners it would be either  
132 beneficial or necessary for each to have some isolation from each other. This  
133 could be achieved via hawkBit's multi-tenant functionality or via the deployment  
134 of multiple instances of hawkBit. It is likely that both of these options would be  
135 deployed depending on the demands and requirements of the product team or  
136 partner. It is expected that some partners may like to use a deployment server  
137 provided by Apertis or one of it's partners. In this instance multi-tenancy would  
138 make sense. Others may wish to have their own instance, possibly hosted by  
139 themselves, in which case providing a simple way to deploy a hawkBit instance  
140 would be beneficial.

141 Deploying multiple instances of hawkBit using the docker configuration would be  
142 trivial. The multi-tenant configuration requires the authentication mechanism  
143 for accessing the management API, web interface and potentially DDI API to  
144 be multi-tenant aware.

## 145 **Life management of artifacts**

146 The GitLab CI pipeline generally performs at least 2 builds a day, pushing  
147 multiple artifacts for each architecture and version of Apertis. In order to  
148 minimize the space used to store artifacts and so as not to store many defunct  
149 artifacts, they are currently deleted after 7 days.

150 Whilst this approach enables the Apertis project to frequently exercise the arti-  
151 fact upload path and has been adequate for Apertis during it's initial phase, a  
152 more comprehensive strategy will be required for production use. For shipped  
153 hardware, it is unlikely that any units will be updated as frequently. In addi-  
154 tion, depending on the form and function of the device, it may only poll the  
155 infrastructure to check for updates sporadically, either due to the device not  
156 needing to be on or not having access to a network connection capable of reach-  
157 ing the deployment server. Artifacts will needed to be more selectively kept to  
158 ensure that the most up-to-date version is kept available for each device type  
159 and hardware revision. Older artifacts that are no longer the recommended ver-  
160 sion should be safe to delete from hawkBit as no targets should be attempting  
161 to update to them.

## 162 **Platform scalability**

163 hawkBit provides support for clustering to scale beyond the bandwidth that a  
164 single deployment server could handle. The Apertis hawkBit instance is not  
165 expected to need to handle a high level of use, though this may be important to  
166 product teams who might quite quickly have many devices connected to hawkBit  
167 in the field.

## 168 **Recommendation**

### 169 **Server Configuration**

- 170 • The improvements made to the Docker Compose configuration file should  
171 be published either in a publicly visible Apertis repository and/or improve-  
172 ments should be submitted back to the hawkBit project to be included in  
173 the reference Docker configuration.

### 174 **Considering the production workflow**

- 175 • The hawkBit deployment should be updated to use a signed key based  
176 security strategy.
- 177 • `apertis-hawkbit-agent` should be improved to enable authentication via  
178 signed keys.
- 179 • `apertis-hawkbit-agent` should be improved to auto-enroll when the target  
180 device is not already found.

- 181 • `apertis-hawkbit-agent` is currently storing its configuration in `/etc`, this  
182 should be extended to look under `/var` and the default configuration should  
183 be moved there.
- 184 • A mechanism should be added to `apertis-hawkbit-agent` to enable the `con-`  
185 `trollerid` to be generated from supported hardware sources.

## 186 Management UI access

- 187 • The Apertis hawkBit instance should be configured to use the OpenID au-  
188 thentication mechanism, ideally using the same SSO used to authenticate  
189 users for other Apertis resources.

## 190 Enabling device filtering

- 191 • Update `apertis-hawkbit-agent` to set attributes based on information  
192 known about the target device. This should include (where possible):  
193 – Device Architecture  
194 – Device Type  
195 – Device Revision

## 196 Provisioning for multiple product teams or partners

- 197 • Apertis does not have a direct need for a multi-tenant deployment nor  
198 for multiple deployments. Investigate and document what's involved for  
199 setting up a multi-tenanted installation.

## 200 Life management of artifacts

- 201 • Apertis is developing a base platform to be used by production teams  
202 and thus the images it produces for it's reference hardware needs a subtly  
203 [different scheme](#)<sup>13</sup> from that which would be anticipated to be needed by a  
204 production team. It is therefore recommended that the process removing  
205 old artifacts should adhere to the following rules:  
206 – Retain all point releases for current Apertis releases  
207 – Retain 7 days of daily development builds  
208 – Delete all artifacts for versions of Apertis no longer supported

## 209 Platform scalability

- 210 • At this current point in time we do not feel that investigating platform  
211 scalability has immediate value.

---

<sup>13</sup><https://sjoerd.pages.apertis.org/apertis-website/architecture/long-term-reproducibility/>