



Web engine

1 **Contents**

2 Security maintenance 2
3 Customization 2
4 White-listing and black-listing 2
5 Rendering of non-web documents 2

6 **Scheduled and potential future work** **3**
7 Web runtime 3

8 Apertis provides the GTK port of WebKit as its web engine. To ensure low
9 maintenance effort, no changes are made to the downstream branch, any im-
10 provements should go to the upstream project.

11 **Security maintenance**

12 Like all other major browser engines, the GTK port does not provide long term
13 support, so security maintenance comes down to staying up to date.

14 The general approach Apertis takes is to follow whatever Debian provides. The
15 project may also importing a new upstream release that has not been made
16 available in Debian yet if an important fix is available.

17 **Customization**

18 Apertis has made a decision to not perform customizations to the engine with
19 the goal of keeping maintenance efforts to a minimum. Whenever a feature is
20 desired, it should be proposed and contributed directly upstream.

21 **White-listing and black-listing**

22 There has been interest in maintaining a black-list of web applications (or pages)
23 that misbehaved. That would be for the case in which the browser gets killed
24 because it stopped responding and the scripts watchdog was not able to restore
25 it to working, so that those web apps or pages are not loaded automatically
26 upon startup causing the browser to go unresponsive again.

27 [Web](https://wiki.gnome.org/Apps/Web)¹ (codename [Epiphany](https://wiki.gnome.org/Apps/Web)²), the GNOME web browser maintains a session file
28 that stores information about all loaded pages, such as title, URL, and whether
29 they are currently loading or not. If Web is quit unexpectedly, it will refuse
30 to load any pages that were marked as still loading automatically. This same
31 approach could be used by the Apertis web browser to not load those pages
32 automatically or to create a blacklist database.

33 The white-list, on the other hand, would be used to enable applications to use
34 lots of resources for a long time without getting killed by this infrastructure

¹<https://wiki.gnome.org/Apps/Web>
²<https://wiki.gnome.org/Apps/Web>

35 in what could be considered a false positive. A white-list can easily be imple-
36 mented, keeping a list of applications that are allowed to go over the limits
37 should be enough.

38 **Rendering of non-web documents**

39 Several kinds of documents that are not strictly web documents are available
40 on web sites for viewing and download. Some of these types of documents, such
41 as PDFs, have become so common that some browsers embed a viewer.

42 WebKit itself does not have support for rendering those documents and the We-
43 bView widget provided by WebKitGTK does not support any kind of custom
44 rendering. Applications and browsers that use the engine can also embed dif-
45 ferent widget alongside the WebView if they would like to allow viewing PDFs
46 and other kinds of documents on the same user interface.

47 **Scheduled and potential future work**

48 **Web runtime**

49 There is interest in providing developers with a way to write applications using
50 web technologies for Apertis. While this is out of the scope of this design, a
51 small description of existing technologies and how they can be applied follows.
52 Collabora can help in the future with a more detailed analysis of what works
53 needs doing, specification and development of a solution.

54 A runner for web applications would ideally create a process for each separate
55 application that will be executed, and use application-specific locations for stor-
56 ing data such as caches and the databases for the features described above –
57 meaning it would not require any kind of special privilege, it would be a reg-
58 ular application. This also means permissions and resource limits can be set
59 individually also for web applications.

60 The fact that more than one process would be executed does not mean a lot
61 of memory overhead, since shared libraries are only loaded in memory once for
62 all processes that use them. This would also have several advantages such as
63 making managing applications permissions easier, and avoiding one application
64 interfering with others.

65 Collabora believes that the best way to provide platform APIs to the web appli-
66 cations would be through custom bindings for relevant interfaces. That ensures
67 the APIs will feel native to the JavaScript environment, and will significantly
68 reduce the attack surface presented to pages, compared to a solution that makes
69 all native libraries available.