



Application entry points

1 Contents

2	Requirements	2
3	Security and privacy considerations	3
4	Menu entries	4
5	Agents	4
6	Non-requirements	4
7	Recommendation	4
8	App identification	4
9	Desktop entries	5
10	Simple applications (one entry point)	9
11	Services	10
12	Entry points which do not appear in the menus	10
13	Multiple-view applications	10
14	Appendix: GSettingsSchema-based entry point registration prior to	
15	October 2015	12
16	Appendix: other approaches to multiple-view applications	14
17	One Desktop Action per view	15
18	One Apertis-specific menu entry per view	16

19 Requirements

20 [Application bundles](#)¹ may contain *application entry points*, which are any of
21 these things:

- 22 • a [graphical program](#)² that would normally appear in a menu
- 23 • a graphical program that would not normally appear in a menu, but can
24 be launched in some other way, for example as a [content-type handler](#)³
- 25 • a [user service](#)⁴ that starts during device startup
- 26 • a user service that is started on-demand

27 Desktop environments provide metadata about these programs so that they can
28 be launched.

29 At least the following use-cases exist:

- 30 • mildenhall-launcher displays a categorized menu of user-facing programs.
31 Typical graphical programs such as the Rhayader web browser must ap-
32 pear here, with a name and an icon.
- 33 • It must be possible to translate the name into multiple languages, with a
34 default (international English) name used for languages where there is no
35 specific translation.
- 36 • Different manufacturers' launcher implementations might have a different
37 taxonomy of categories for programs.

¹<https://em.pages.apertis.org/apertis-website/glossary/#application-bundles>

²<https://em.pages.apertis.org/apertis-website/glossary/#graphical-program>

³https://em.pages.apertis.org/apertis-website/concepts/content_hand-over/

⁴<https://em.pages.apertis.org/apertis-website/glossary/#user-service>

- 38 • If two graphical programs have the same user-facing name, it might be
39 useful to be able to provide a longer distinguishing name. For example,
40 if both Chrome and Firefox are installed, they might be called “Firefox
41 Browser” and “Chrome Browser”, but if only one is installed, it might
42 simply be called “Browser”.
- 43 • Certain graphical programs should be hidden from the menu, but treated
44 as a first-class program during user interaction. As of October 2015, the
45 Canterbury application manager has hard-coded special cases for various
46 removable storage browsing applications; an improved metadata format
47 would allow these special cases to be generalized.
- 48 • Some graphical programs present multiple *views* which may appear sep-
49 arately in menus, but are all implemented in terms of the same running
50 process. For example, the Frampton audio player appears in the menu
51 three times, as “Albums”, “Artists” and “Songs”. However, ideally there
52 would only be one Frampton HMI process at any given time, even if the
53 user switches between views.
- 54 • Some programs should be started during device startup or user login.
- 55 • In the SDK images, Apertis applications and services should not necessar-
56 ily be listed in the XFCE menus, and XFCE applications should not be
57 listed in the “device simulator”.

58 Security and privacy considerations

59 The list of installed store application bundles in /Applications is considered to
60 be private information, for several reasons:

- 61 • the general operating principle for Apertis’ app framework is that apps
62 must not affect each other, except where given permission to interact,
63 ensuring “loose coupling” between apps
- 64 • the presence of certain app bundles might be considered to be sensitive
65 (for example, app bundles correlated with political or religious views)
- 66 • the complete list could be used for user fingerprinting, for example guessing
67 that users of an online service share a device by observing that they have
68 the same list of app-bundles

69 The list of installed entry-points is almost equivalent to the list of store applica-
70 tion bundles and has similar considerations. However, some components cannot
71 work without a list of store application bundles, or a list of their entry points.
72 This leads to some privacy requirements:

- 73 • Certain platform components such as the Canterbury app manager,
74 the Didcot content handover service, and the mildenhall-launcher app-
75 launching HMI require the ability to list store application bundles and/or
76 their entry points. They must be able to do so.
- 77 • Store applications with special permissions might also be allowed to list
78 store application bundles and/or their entry points.
- 79 • Store applications may list the entry points that advertise a particular

80 *public interface*, as defined in the [Interface discovery](#)⁵ design.
81 • Store applications without special permissions must not be able to enu-
82 merate store application bundles that do not contain an entry point ad-
83 vertising a public interface, either directly or by enumerating entry points
84 and inferring the existence of a bundle from its entry points.

85 Unlike store application bundles, we suggest that the list of installed built-in
86 application bundles in `/usr/Applications` should *not* be considered to be private.
87 This list will be the same for every instance of the same platform image, so an
88 application author could learn this list by querying the platform image variant
89 and version, then matching that to a pre-prepared list of application bundles
90 known to exist in their own copy of the same image. Conversely, because this
91 list is the same for every instance of the same platform image, it is not useful
92 for user fingerprinting.

93 **Menu entries**

94 Optionally, a single entry point may be specified to provide an icon for presen-
95 tation in the application launcher. If no icon is presented it won't be obvious
96 to the user that they have the application installed, so the application store
97 screening process should carefully consider whether an application should be
98 allowed to install services and type handlers with no icon for the launcher.

99 The Applications concept design has historically assumed that application bun-
100 dles should be constrained to contain at most one menu entry. However, one
101 of the reference app-bundles developed as part of Apertis (the [Frampton media](#)
102 [player](#)⁶) has multiple menu entries, so this document has assumed that this
103 constraint is no longer desired.

104 **Agents**

105 Agents should be specified as entry points, with a localized list of names for the
106 agent, along with the location of the executable file to launch. Since agents can
107 be long running and have an impact on device performance, any application
108 with an agent should also set the `agent` [permission](#)⁷ so the user can choose not
109 to install the application.

110 **Non-requirements**

111 [System services](#)⁸ are outside the scope of this design.

⁵https://em.pages.apertis.org/apertis-website/concepts/interface_discovery/

⁶<https://gitlab.apertis.org/appfw/frampton/tree/v0.6.1/scripts>

⁷[applications.md#permissions](#)

⁸<https://em.pages.apertis.org/apertis-website/glossary/#system-service>

112 Recommendation

113 The [Apertis Application Bundle Specification](#)⁹ describes the fields that can ap-
114 pear in application entry points and are expected to remain supported long-
115 term. This document provides rationale for those fields, suggested future direc-
116 tions, and details of functionality that is not necessarily long-term stable.

117 App identification

118 Each built-in or store application bundle has a *bundle ID*, which is a [reversed](#)
119 [domain name](#)¹⁰ such as `org.apertis.Frampton`.

120 Each entry point within an application bundle has an *entry point ID*, which is
121 a reversed domain name such as `org.apertis.Frampton.Agent`.

122 For simple bundles with a single entry point, the bundle ID and the entry point
123 ID should be equal.

124 For more complex bundles with multiple entry points, the entry point ID should
125 *start with* the bundle ID, but may have additional components.

126 All names should be allocated in a namespace controlled by the author of the
127 bundle — in particular, Apertis applications should be in `org.apertis`. Sam-
128 ple code that is not intended to be used in production should be placed in
129 `com.example`, with `org.example` and `net.example` also available for code samples
130 that need to demonstrate the interaction between multiple namespaces (we pre-
131 fer `com.example`, as a hint to developers that reversed domain names do not
132 always start with “org”).

133 Desktop entries

134 Each Apertis *application entry point* is represented by a standard freedesk-
135 top.org [Desktop Entry](#)¹¹ (a `.desktop` file in `XDG_DATA_DIRS/applications`). The
136 desktop file must be named using the entry point ID, so `org.apertis.Frampton.Agent`
137 would have `org.apertis.Frampton.Agent.desktop`.

138 The [localestring](#)¹² mechanism is used for translated strings.

139 Built-in application bundles install their desktop files in `${prefix}/share/applications`,
140 which expands to `/usr/Applications/${bundle_id}/share/applications`. They
141 also install symbolic links in `/usr/share/applications` pointing to the real files.
142 It is technically possible for any process to read this location.

143 Store applications install their desktop files in `${prefix}/share/applications`,
144 which expands to `/Applications/${bundle_id}/share/applications`. The app in-
145 staller is responsible for creating symbolic links in `/var/lib/apertis_extensions/applications`

⁹<https://em.pages.apertis.org/apertis-website/architecture/bundle-spec/>

¹⁰https://en.wikipedia.org/wiki/Reverse_domain_name_notation

¹¹<http://standards.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html>

¹²<https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#localized-keys>

146 pointing to the real files. Only processes with appropriate permissions are
147 allowed to read these locations.

148 Apertis applications must have the `X-Apertis-Type` key in their metadata, so that
149 they will be listed in Apertis. They should usually also have `OnlyShowIn=Apertis`;
150 so that they do not appear in the XFCE desktop environment menu in SDK
151 images.

152 The value of the `Exec` key must start with an absolute path to the executable
153 below `$_prefix`. This ensures that the application framework can detect which
154 app-bundle the executable belongs to.

155 Entry points that would not normally appear in a menu, including all back-
156 ground services (agents), should have `NoDisplay=true`.

157 The `Interfaces` key is used for [Interface discovery](#)¹³. In particular, the following
158 interfaces are defined:

- 159 • `org.apertis.GlobalSearchProvider`: Indicates that the application is a
160 global search provider, equivalent to the `supports-global-search` schema
161 entry.

162 The standard `MimeType` key controls the possible [content-type and URI-scheme](#)
163 [associations](#)¹⁴. For example, `x-scheme-handler/http` is used in desktop environ-
164 ments such as GNOME to designate an application as capable of acting as a
165 general-purpose web browser, and we will do the same here. The Didcot ser-
166 vice mediates applications' access to this information; for example, it may set
167 priorities or ignore certain applications or associations altogether.

168 Services that parse desktop files should use [the implementation in GLib](#)¹⁵, or
169 an Apertis-specific API built on top of that implementation.

170 Additional recommended keys

171 The following additional keys are defined in the `[Desktop Entry]` group.

- 172 • `X-Apertis-ParentEntry` (string): For situations where multiple menu entries
173 start the same program in different modes, all but one of those menu en-
174 tries set `X-Apertis-ParentEntry` to the entry point ID of the remaining menu
175 entry. See [Multiple-view applications](#) and the `[D-Bus Activation][Bundle`
176 `spec D-Bus activation]` section of the Apertis Application Bundle Specifi-
177 cation.
- 178 • `X-Apertis-ServiceExec` (string): A command-line similar to `Exec` that starts
179 the entry point in the background, without implicitly *activating* it (caus-
180 ing it to show a window) if it is a graphical program. For example, entry
181 points that use `GApplication` will usually use the same executable as for

¹³https://em.pages.apertis.org/apertis-website/concepts/interface_discovery/

¹⁴https://em.pages.apertis.org/apertis-website/concepts/content_hand-over/

¹⁵<https://gitlab.gnome.org/GNOME/glib/-/blob/main/gio/gdesktopappinfo.c#L2007>

182 Exec here, but add the `--gapplication-service` option to it. See the [D-Bus
183 Activation][Bundle spec D-Bus activation] section of the Apertis Applica-
184 tion Bundle Specification.

- 185 • `X-GNOME-FullName` (localestring): The human-readable full name of the ap-
186 plication, such as `Rhayader Web Browser`. This key is already used by the
187 GLib library, and by desktop environments based on it (such as GNOME).
188 Like `Name`, this is a “localestring”: non-English versions can be provided
189 with syntax like `X-GNOME-FullName[fr]=Navigateur Web Rhyader`.

190 Potential future keys

191 The following additional keys have been proposed for the [Desktop Entry] group.

- 192 • `X-Apertis-BandwidthPriority` (string): Bandwidth priority, currently cho-
193 sen from `highest`, `high`, `normal`, `low` or `lowest`. As a future extension, nu-
194 meric priorities could be added, with those strings mapped to reasonable
195 values.

196 Audio roles

197 Requirements-gathering for the audio manager is ongoing. An `x-`
198 `Apertis-AudioRole` key was initially proposed, but it seems likely that
199 support for specifying a default audio role for PulseAudio streams
200 that do not specify one will be moved from entry points into [appli-
201 cation bundle metadata](#)¹⁶.

202 The audio role should have one of the [well-known media roles defined by
203 PulseAudio](#)¹⁷.

204 Additionally, Apertis defines the following roles. Their semantics are not clear,
205 and they should be clarified or deprecated.

- 206 • `none`
- 207 • `interrupt`
- 208 • `record` (possibly the same thing as PulseAudio’s `production`, denoting an
209 application that creates or edits audio files, such as a sound recorder)
- 210 • `external`
- 211 • `unknown`

212 Additional provisional/deprecated keys

213 The following provisional keys are defined in the [Desktop Entry] group, but are
214 anticipated to be superseded, adjusted or redefined in future.

- 215 • `X-Apertis-Type` (string): The application type, chosen from `application`,
216 `service`, `ext-app`, `agent-service`, `startup-application`. Applications with

¹⁶[application-bundle-metadata.md](#)

¹⁷<http://www.freedesktop.org/wiki/Software/PulseAudio/Documentation/Developer/Clients/ApplicationProperties/>

217 no `X-Apertis-Type` are not currently run or displayed in Apertis. This
 218 should eventually be replaced with a set of boolean flags describing specific
 219 behaviours, such as “start immediately” and “is expected to display a
 220 window”; these could either be flags in the file, or indicated in another
 221 appropriate way, for example a symbolic link in `/etc/xdg/autostart` for
 222 applications and services that should be started immediately.

- 223 • `X-Apertis-CategoryLabel` (string; this would normally be a localestring, but
 224 the current mildenhall-launcher relies on specific string values for category
 225 labels, so translating it is not useful): The name of the menu category.
 226 This will be implemented in the short term to keep the current version of
 227 Mildenhall-Launcher operational, but should be considered to be depre-
 228 cated. Instead, launchers should parse the standard `Categories` key, which
 229 contains a list of standardized machine-readable categories with the pos-
 230 sibility to add Apertis-specific extensions, and translate those into the
 231 categories required by the desired UX.
- 232 • `X-Apertis-CategoryIcon` (string): The short name of an icon for the cate-
 233 gory, such as `icon_settings_AC`. In the short term, Canterbury translates
 234 this to `/icon_settings_AC.png` to keep the current version of Mildenhall-
 235 Launcher operational. Like `X-Apertis-CategoryLabel`, this should be con-
 236 sidered to be deprecated; instead, the launcher should determine an icon
 237 name from the standard `Categories` key.
- 238 • `X-Apertis-BackgroundState` (string): What will happen to the application
 239 when it is placed in the background: `running` (i.e. don’t kill), `stopped` (i.e.
 240 pause the process), `killed` (i.e. kill the process). This key and its values
 241 should ideally be replaced with something that more obviously describes
 242 an action rather than a state, such as `kill`, `pause`, `continue`.
- 243 • `X-Apertis-DataExchangeRules` (string): This appears to be something to do
 244 with Didcot, but its semantics are unclear. The only known example is
 245 `default-data`. It should be clarified or dropped.
- 246 • `X-Apertis-ManifestUrl` (string): This appears to be intended to point to
 247 the JSON manifest for the app bundle, but in the majority of the apps
 248 that are currently implemented, it points to a nonexistent XML file, or
 249 to the GSettings schema in which it is defined. It should be clarified or
 250 dropped.
- 251 • `X-Apertis-SplashScreen` (string): None of the current app bundles have
 252 this, and it is unclear what its value is meant to be. It is currently passed
 253 to the compositor via a D-Bus method call.

254 Transitional considerations

255 In addition to `/var/lib/apertis_extensions/applications`, Canterbury reads store
 256 app bundles’ entry points from `/var/lib/MILDENHALL_extensions/applications`
 257 and `/var/lib/SAC_extensions/applications`, which are two older names for the
 258 same thing. We should remove that feature when everything has migrated to
 259 `/var/lib/apertis_extensions/applications`.

260 Canterbury currently has special handling for the executable’s arguments:

- 261 • An argument named exactly `url` is assumed to be followed by a place-
262 holder; that placeholder is replaced by the actual URL if the application
263 is to be launched with a URL argument. In the short term, this will be pre-
264 served. In the longer term, Canterbury and applications should migrate
265 to [the standard `%u`, `%f`, `%U`, `%F` placeholders] [Desktop Entry placeholders](#)¹⁸
266 for a URL, filename, list of URLs or list of filenames respectively.
- 267 • An argument named exactly `app-name` is assumed to be followed by a place-
268 holder; that placeholder is replaced by the *entry point ID*. In the short
269 term, this will be preserved. In the longer term, this should be dropped;
270 applications should know their own entry point IDs.
- 271 • An argument named exactly `play-mode` is assumed to be followed by a
272 placeholder; that placeholder is replaced by `play` or `stop`. In the short
273 term, this will be preserved. In the longer term, media player applications
274 should implement [Desktop Entry actions](#)¹⁹ instead.

275 There is currently special handling for several arguments with value exactly `###`
276 `UNKNOWN ###`. In the long term this should be removed.

277 In the long term, the category should be replaced by the standard `Categories`
278 key, preferably with values chosen from the [XDG Desktop Menu specification](#)²⁰.
279 This would allow for variants that do not use precisely the same taxonomy of
280 applications as `mildenhall-launcher`; because `Categories` is a list, the launcher
281 may use fine-grained categories if desired, falling back to more general top-
282 level categories such as `AudioVideo` if it does not understand any more specific
283 category.

284 The application launcher HMI should translate these categories into whatever
285 was specified by the variant’s UX designer; for example, `mildenhall-launcher`
286 would translate `Video` to “Video & TV”, `Office` to “Productivity”, and `Maps` to
287 “Travel”. The application launcher HMI should also be responsible for presenta-
288 tional logic such as displaying “Travel” as “T R A V E L” if desired.

289 **Features with no direct replacement**

290 `env-key-value-pair` in the `GSettings` schemata does not currently appear to be
291 used. We recommend removing this feature: application bundles should nor-
292 mally be written to not need a special environment. If they do need special
293 environment variables, the desktop file could specify a shell script as its `Exec`
294 program, with that shell script setting appropriate environment variables and
295 then `execing` the real binary.

¹⁸<http://standards.freedesktop.org/desktop-entry-spec/latest/ar01s06.html#exec-variables>

¹⁹<http://standards.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#extra-actions>

²⁰<http://standards.freedesktop.org/menu-spec/latest/apa.html>

296 `tile-thumbnails` in the `GSettings` schemata does not currently appear to be used.
297 A replacement can be added when the requirements are more clear.

298 **Simple applications (one entry point)**

299 This is the simple case where an entry point has one “view”, for example the
300 Rhayader web browser.

301 We install symlinks in `/usr/share/applications` (for built-in app bundles) or
302 `/var/lib/apertis_extensions/applications` (for store app bundles) pointing to the
303 real file in `{/usr,}/Applications/${bundle_id}/share/applications`, with content
304 similar to this.

```
305 # /usr/share/applications/org.apertis.Rhayader.desktop
306 [Desktop Entry]
307 Type=Application
308 Name=Rhayader
309 GenericName=Browser
310 X-GNOME-FullName=Rhayader Browser
311 Exec=/usr/Applications/org.apertis.Rhayader/bin/rhayader %U
312 Path=/usr/Applications/org.apertis.Rhayader
313 X-Apertis-Type=application
314 X-Apertis-InternetPriority=normal
315 Categories=Network;WebBrowser;
316 MimeType=text/html;x-scheme-handler/http;x-scheme-handler/https;
317 Icon=applications-internet
```

318 **Services**

319 Services are the same as applications (in particular, they have `Type=Application`),
320 except for these special cases:

- 321 • they have `NoDisplay=true` to hide them from the menus
- 322 • the `X-Apertis-Type` is `service` OR `agent-service`

323 **Entry points which do not appear in the menus**

324 Some bundles might have an entry point that exists only to be started as a
325 side-effect of other operations, for instance to [handle URIs and content-types](#)²¹.
326 Those entry points would have `NoDisplay=true` to hide them from the menus;
327 that is the only difference.

328 **Multiple-view applications**

329 Some bundles have more than one entry in the system menus; the example we
330 know about is Frampton. We propose to represent these with one `.desktop` file
331 per menu entry.

²¹https://em.pages.apertis.org/apertis-website/concepts/content_hand-over/

332 In this model, each menu entry is a `.desktop` file. Frampton would install
333 `org.apertis.Frampton.Artists.desktop`, `org.apertis.Frampton.Songs.desktop`
334 and `org.apertis.Frampton.Albums.desktop`. In addition, it would install
335 `org.apertis.Frampton.desktop` with `NoDisplay=true`.

336 The running instance of Frampton would always identify itself as `org.apertis.Frampton`,
337 and the other three `.desktop` files use `X-Apertis-ParentEntry=org.apertis.Frampton`
338 to link them to that name.

339 When using [D-Bus activation]Desktop Entry D-Bus Activation²² for ap-
340 plications (which is recommended), Frampton would have separate D-
341 Bus `.service` files for all four names, would take all four bus names
342 and their corresponding object paths at runtime, and would export the
343 `org.freedesktop.Application` API at all four paths; but all of them would have
344 `SystemService=org.apertis.Frampton.service` to ensure that only one activation
345 occurs. The `Activate`, `Open` or `ActivateAction` method on each bus name would
346 open the relevant view.

347 The result would look something like this:

```
348 # org.apertis.Frampton.desktop
349 [Desktop Entry]
350 Type=Application
351 Name=Frampton
352 GenericName=Audio Player
353 X-GNOME-FullName=Frampton Audio Player
354 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton %F
355 Path=/usr/Applications/org.apertis.Frampton
356 X-Apertis-Type=application
357 Categories=Audio;Player;Music;
358 MimeType=audio/mpeg;
359 NoDisplay=true;
360 Icon=music
361 X-Apertis-ServiceExec=/usr/Applications/org.apertis.Frampton/bin/frampton --
362 gapplication-service

363 # org.apertis.Frampton.Artists.desktop
364 [Desktop Entry]
365 Type=Application
366 Name=Frampton — Artists
367 GenericName=Artists
368 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton --artists
369 Path=/usr/Applications/org.apertis.Frampton
370 X-Apertis-Type=application
371 Categories=Audio;Player;Music;
372 Icon=music-artist
```

²²<http://standards.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#dbus>

```

373 X-Apertis-ParentEntry=org.apertis.Frampton
374 # org.apertis.Frampton.Albums.desktop
375 [Desktop Entry]
376 Type=Application
377 Name=Frampton — Albums
378 GenericName=Albums
379 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton --albums
380 Path=/usr/Applications/org.apertis.Frampton
381 X-Apertis-Type=application
382 Categories=Audio;Player;Music;
383 Icon=music-album
384 X-Apertis-ParentEntry=org.apertis.Frampton

385 # org.apertis.Frampton.Songs.desktop
386 [Desktop Entry]
387 Type=Application
388 Name=Frampton — Songs
389 GenericName=Songs
390 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton --songs
391 Path=/usr/Applications/org.apertis.Frampton
392 X-Apertis-Type=application
393 Categories=Audio;Player;Music;
394 Icon=music-track
395 X-Apertis-ParentEntry=org.apertis.Frampton

```

396 **Appendix: GSettingsSchema-based entry point registra-** 397 **tion prior to October 2015**

398 As of early October 2015, Canterbury uses GSettings schemata for entry point
399 registration. This is not an intended use of GSettings — the existence of an
400 entry point is not a setting — and it should be avoided.

401 Canterbury reads the schemata from the default system paths, and from a con-
402 figurable path (`p_app_manager_get_store_apps_schema_path()`) which in practice
403 resolves to `/Applications/System/registry`. For each schema in the path, if the
404 name does not start with either `com.app` (this prefix is actually configurable,
405 project-domain) or `org.secure_automotive_cloud.service`, then the schema is ig-
406 nored.

407 *Proposed replacement: Canterbury reads desktop files from at least*
408 */var/lib/apertis_extensions/applications and /usr/share/applications, and*
409 *may read additional locations if desired. This should be done by setting*
410 *Canterbury's XDG_DATA_DIRS to include at least /var/lib/apertis_extensions and*
411 */usr/share.*

412 Canterbury reads the following keys, each with a corresponding constant such
413 as `APP_NAME` except where noted:

- 414 • `app-name` (`pAppName`): A string: entry point ID, such as `frampton`, `Frampton-`
415 `Agent`. *Proposed replacement: the name of the `.desktop` file.*
- 416 • `background-state` (`uinBkgState`): One of { `running`, `stopped`, `killed`, `unknown`
417 }. *Proposed replacement: `X-Apertis-BackgroundState`*
- 418 • `working-directory` (`pWorkingDirectory`): A string: the app's initial working
419 directory, which in practice must be in its directory `/usr/Applications/xyz`
420 for built-in apps (Ribchester assumes this, and uses it to create the app's
421 storage during first-boot). *Proposed replacement: the standard `Path` key.*
- 422 • `exec-path` (`pExecutablePath`): A string: the executable. *Proposed replace-*
423 *ment: the first word of the standard `Exec` key.*
- 424 • `exec-type` (`uinExecutableType`): One of { `application`, `service`, `ext-`
425 `application` (or sometimes `ext-app`, depending on project), `agent-service`,
426 `unknown` }. *Proposed replacement: `X-Apertis-Type`.*
- 427 • `exec-args` (`pExecutableArgv`): An array of (string, string) pairs which
428 are flattened into a single list for `exec()`, for example `[('app-name',`
429 `'AudioPlayer')`, `('menu-entry', 'A R T I S T S')`, `('url', ' ')]`
430 turns into executing the equivalent of the Python code `subpro-`
431 `cess.call(['/usr/Applications/frampton/bin/frampton', 'app-name',`
432 `'AudioPlayer', 'menu-entry', 'A R T I S T S', 'url', ' '])`. *Proposed*
433 *replacement: the standard `Exec` key, except for its first word.*
- 434 • `internet-bw-prio` (`uinInternetBandwidthPriority`): One of { `highest`, `high`,
435 `mid`, `low`, `lowest`, `unknown` } or unspecified. *Proposed replacement: `X-`*
436 *`Apertis-BandwidthPriority`. Additionally, we recommend accepting normal*
437 *as a synonym for `mid`.*
- 438 • `splash-screen` (`pSplashScreen`): A string. No application specifies this, so
439 we do not know what its purpose is. *Proposed replacement: `X-Apertis-`*
440 *`SplashScreen`, or remove the feature.*
- 441 • `audio-resource-type` (`CANTERBURY_AUDIO_RESOURCE_TYPE`, `uinAudioResource-`
442 `Type`): A `CanterburyAudioType`. *Proposed replacement: use `PulseAudio`*
443 *stream roles.*
- 444 • `audio-channel-name` (`pAudioChannelName`): A string: the name of the audio
445 channel. *Proposed replacement: make the audio manager derive the bundle*
446 *ID from the `AppArmor` profile in a way that cannot be faked by a malicious*
447 *app-bundle.*
- 448 • `audio-resource-owner` (`CANTERBURY_AUDIO_RESOURCE_OWNER`, `pAudioResource-`
449 `Owner`): A string: the entry point ID of the entry point that will generate
450 audio on behalf of this HMI. *Proposed replacement: make the audio*
451 *manager derive the bundle ID from the `AppArmor` profile in a way that*
452 *cannot be faked by a malicious app-bundle.*
- 453 • `category` (`pCategory`): A string: the displayed name of the category. *Pro-*
454 *posed replacement: `X-Apertis-CategoryLabel` in the short term, `Categories`*
455 *in the longer term.*
- 456 • `category-icon` (`pCategoryIcon`): A string: the name of the category icon,
457 of the form `/icon.png`, which appears to be relative to the launcher's data
458 directory. *Proposed replacement: `Icon`, changing the value to be defined to*
459 *be found via the `freedesktop.org` icon theme specification.*

- 460 • `env-key-value-pair` (`ENV_KEY_VALUE`, `pEnvKeyValuePair`): An array of strings,
461 of even length: the environment of the subprocess. *Proposed replacement:*
462 *remove.*
- 463 • `window-name` (`APP_WIN_NAME`, `pAppWinName`): A string: the name of the win-
464 dow that this HMI is expected to map. *Proposed replacement: make the*
465 *compositor derive the bundle ID from the AppArmor profile in a way that*
466 *cannot be faked by a malicious app-bundle.*
- 467 • `application-entry-names` (`APP_ENTRY_NAME_LIST`, `pApplicationEntryName`): An
468 array of strings. Each one is the title of a quick-menu (right panel) entry
469 in `mildenhall-launcher`. The main-menu (left panel) entry is taken from
470 `category` and `category-icon`. *Implemented replacement: one desktop file*
471 *per entry point, and use its `X-GNOME-FullName`, `GenericName` and/or `Name`.*
472 *The ability to have more than one menu entry per application is replaced*
473 *by `X-Appertis-ParentEntry`.*
- 474 • `application-entry-icons` (`APP_ENTRY_ICON_LIST`, `pApplicationEntryIcon`): An
475 array of strings: `file:///` URLs to icons, in the same order as `application-`
476 `entry-names`. *Implemented replacement: Icon may name either an icon in*
477 *the icon theme, or a `file:///` URL.*
- 478 • `tile-thumbnails` (`APP_ENTRY_TILE_LIST`, `pApplicationTileThumbnail`): An ar-
479 ray of strings that represent home-screen tiles in some unspecified way;
480 we do not have any examples to use for reference. *Proposed replacement:*
481 *behave as though no application has a home-screen tile, and design home-*
482 *screen tiles separately.*
- 483 • `manifest-url` (`MANIFEST_FILE_URL`, `pAppManifestUrl`): A string representing
484 the manifest in some way. *Proposed replacement: `X-Appertis-ManifestUrl`,*
485 *or remove; services should find desktop files for entry points in the standard*
486 *way, and should find manifests for app-bundles by looking in well-known*
487 *locations for files whose names are based on the bundle ID.*
- 488 • `app-settings-icon` (`pAppSettingsIcon`): A string representing an icon used
489 for settings. *Proposed replacement: icon from [application bundle meta-](#)*
490 *[data](#)²³.*
- 491 • `app-settings-name` (`pAppSettingsName`): A string representing a label used
492 for settings? *Proposed replacement: name from [application bundle meta-](#)*
493 *[data](#)²⁴.*
- 494 • `app-settings-path` (`pAppSettingsPath`): A string representing a GSettings
495 hierarchy used for settings. *Implemented replacement: the settings schema*
496 *(if any) whose name matches the bundle ID appears in the system prefer-*
497 *ences UI; other schemas do not appear.*
- 498 • `mime-type` (`pMimeType`): An array of strings representing content-types
499 that this application can handle, and/or pseudo-content-types such as
500 `mt_app_settings`. *Implemented replacement: `MimeType` for content-type and*
501 *URL-scheme handlers; Interfaces to discover other functionality.*
- 502 • `mime-list` (`pMimeTypeList`): An array of strings representing some facet of con-

²³[application-bundle-metadata.md](#)

²⁴[application-bundle-metadata.md](#)

503 tent type handling, with values such as `url`, `audio/mpeg` and `launch`. *Proposed replacement: discover feature support with Interfaces.*

504

- 505 • `data-exchange-rules` (`DATA_EXCHANGE_FILE`, `pDataExchangeFile`): A string
- 506 which has something to do with data exchange, with the only known
- 507 value being `default-data`. *Proposed replacement: none.*
- 508 • `supports-global-search` (`SUPPORT_GLOBAL_SEARCH`, `bSupportsGlobalSearch`):
- 509 A boolean value indicating support for acting as a global search
- 510 provider. *Proposed replacement: if this would have been true, then*
- 511 *`org.apertis.GlobalSearchProvider` appears in Interfaces.*

512 Appendix: other approaches to multiple-view applications

513 We considered some other approaches to this feature.

514 One Desktop Action per view

515 In this model, each entry point (application or service) is a `.desktop` file. Frampton would install `org.apertis.Frampton.desktop`, with contents something like this:

```
518 # org.apertis.Frampton.desktop
519 [Desktop Entry]
520 Type=Application
521 Name=Frampton
522 GenericName=Audio Player
523 X-GNOME-FullName=Frampton Audio Player
524 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton %F
525 Path=/usr/Applications/org.apertis.Frampton
526 X-Apertis-Type=application
527 X-Apertis-AudioRole=music
528 X-Apertis-AudioChannelName=org.apertis.Frampton.Agent
529 Categories=Audio;Player;Music;
530 MimeType=audio/mpeg;
531 NoDisplay=true;
532 Actions=albums;artists;songs;
533 Icon=music
534
535 [Desktop Action artists]
536 Name=Artists
537 Icon=music-artist
538 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton --artists
539 X-Apertis-ShowInMenu=true
540
541 [Desktop Action albums]
542 Name=Albums
543 Icon=music-album
544 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton --albums
```

```

545 X-Apertis-ShowInMenu=true
546
547 [Desktop Action songs]
548 Name=Songs
549 Icon=music-track
550 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton --songs
551 X-Apertis-ShowInMenu=true
552
553 # this is *not* a "quick menu" entry
554 [Desktop Action shuffle]
555 Name=Shuffle All
556 Icon=music-shuffle
557 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton-control --shuffle-
558 all

```

559 The Desktop Entry Specification specifies that application launchers should
560 present [desktop actions](#)²⁵ to the user within the context of an application,
561 for instance as a submenu, but that isn't how the UX of `mildenhall-launcher`
562 works. We therefore use `X-Apertis-ShowInMenu` to indicate that these particular
563 desktop actions should be made available to the user even though their parent
564 `org.apertis.Frampton` is not.

565 This could be combined with desktop actions as specified in the Desktop Entry
566 Specification if desired; those desktop actions would simply omit `X-Apertis-`
567 `ShowInMenu`. For example, if it was desirable for a long press on Frampton's
568 menu entries to result in a menu of actions such as "shuffle all", "import from
569 USB drive", "buy music", then those could be represented as desktop actions.

570 **One Apertis-specific menu entry per view**

571 This model is similar to the one with desktop actions, but it acknowledges that
572 desktop actions were not really designed to work that way, and uses Apertis-
573 specific syntax inspired by desktop actions instead:

```

574 # org.apertis.Frampton.desktop
575 [Desktop Entry]
576 Type=Application
577 Name=Frampton
578 GenericName=Audio Player
579 X-GNOME-FullName=Frampton Audio Player
580 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton %F
581 Path=/usr/Applications/org.apertis.Frampton
582 X-Apertis-Type=application
583 X-Apertis-AudioRole=music
584 X-Apertis-AudioChannelName=org.apertis.Frampton.Agent

```

²⁵<http://standards.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#extra-actions>


```
585 Categories=Audio;Player;Music;
586 MimeType=audio/mpeg;
587 X-Apertis-MenuEntries=albums;artists;songs;
588 Icon=music
589
590 [Apertis Menu Entry artists]
591 Name=Frampton — Artists
592 GenericName=Artists
593 Icon=music-artist
594 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton --artists
595
596 [Apertis Menu Entry albums]
597 Name=Frampton — Albums
598 GenericName=Albums
599 Icon=music-album
600 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton --albums
601
602 [Apertis Menu Entry songs]
603 Name=Frampton — Songs
604 GenericName=Songs
605 Icon=music-track
606 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton --songs
607
608 [Desktop Action shuffle]
609 Name=Shuffle All
610 Icon=music-shuffle
611 Exec=/usr/Applications/org.apertis.Frampton/bin/frampton-control --shuffle-
612 all
```