



Application entry points

Contents

2	Requirements	2
3	Security and privacy considerations	3
4	Menu entries	3
5	Background Services	3
6	Non-requirements	4
7	Recommendation	4
8	App identification	4
9	Desktop entries	4
10	Simple applications (one entry point)	5
11	Entry points which do not appear in the menus	6
12	Multiple-view applications	6

Requirements

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

- a [graphical program](#)² that would normally appear in a menu
- a graphical program that would not normally appear in a menu, but can be launched in some other way, for example as a [content-type handler](#)³

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

At least the following use-cases exist:

- It must be possible to translate the name into multiple languages, with a default (international English) name used for languages where there is no specific translation.
- Different manufacturers'launcher implementations might have a different taxonomy of categories for programs.
- Certain graphical programs should be hidden from the menu, but treated as a first-class program during user interaction.
- Some graphical programs present multiple *views* which may appear separately in menus, but are all implemented in terms of the same running process. For example, an audio player may appear in the menu three times, as "Albums", "Artists"and "Songs". However, ideally there would only be one audio player process at any given time, even if the user switches between views.
- Some programs should be started during device startup or user login.
- In the SDK images, Apertis applications and services should not necessarily be listed in the XFCE menus, and XFCE applications should not be

¹<https://apertis-website-0b3586.pages.apertis.org/glossary/#application-bundles>

²<https://apertis-website-0b3586.pages.apertis.org/glossary/#graphical-program>

³https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_framework/content_hand-over/

38 listed in the “device simulator”.

39 Security and privacy considerations

40 The list of installed store application bundles is considered to be private infor-
41 mation, for several reasons:

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

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

- 54 • Certain platform components such as the app launcher require the ability
55 to list store application bundles and/or their entry points. They must be
56 able to do so.
- 57 • Store applications with special permissions might also be allowed to list
58 store application bundles and/or their entry points.
- 59 • Store applications may list the entry points that advertise a particular
60 *public interface*, as defined in the [Interface discovery](https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_framework/interface_discovery/)⁴ design.
- 61 • Store applications without special permissions must not be able to enu-
62 merate store application bundles that do not contain an entry point ad-
63 vertising a public interface, either directly or by enumerating entry points
64 and inferring the existence of a bundle from its entry points.

65 Menu entries

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

71 Background Services

72 Background services in the current Flatpak distribution model do *not* use entry
73 points; rather, they request the ability to run arbitrary command lines in the

⁴https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_framework/interface_discovery/

74 background at runtime via the [XDG Background portal](#)⁵.

75 Non-requirements

76 [System services](#)⁶ are outside the scope of this design.

77 Recommendation

78 [Application bundle metadata](#)⁷ describes the fields that can appear in application
79 entry points and are expected to remain supported long-term. This document
80 provides rationale for those fields, suggested future directions, and details of
81 functionality that is not necessarily long-term stable.

82 App identification

83 Each built-in or store application bundle has a *bundle ID*, which is a [reversed](#)
84 [domain name](#)⁸ such as `org.apertis.Example`.

85 Each entry point within an application bundle has an *entry point ID*, which is
86 a reversed domain name such as `org.apertis.Example.Entry`.

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

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

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

98 Desktop entries

99 Each Apertis *application entry point* is represented by a standard freedesk-
100 top.org [Desktop Entry](#)⁹ (a `.desktop` file in `XDG_DATA_DIRS/applications`). The
101 desktop file must be named using the entry point ID, so `org.apertis.Example`
102 would have `org.apertis.Example.desktop`.

⁵<https://flatpak.github.io/xdg-desktop-portal/portal-docs.html#gdbus-org.freedesktop.portal.Background>

⁶<https://apertis-website-0b3586.pages.apertis.org/glossary/#system-service>

⁷[application-bundle-metadata.md](#)

⁸https://en.wikipedia.org/wiki/Reverse_domain_name_notation

⁹<http://standards.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html>

103 The `localestring`¹⁰ mechanism is used for translated strings.

104 Apertis applications should usually have `OnlyShowIn=Apertis`; so that they do
105 not appear in the XFCE desktop environment menu in SDK images.

106 The `Interfaces` key is used for [Interface discovery](#)¹¹. In particular, the following
107 interfaces are defined:

- 108 • `org.apertis.GlobalSearchProvider`: Indicates that the application is a
109 global search provider, equivalent to the `supports-global-search` schema
110 entry.

111 The standard `MimeType` key controls the possible [content-type and URI-scheme](#)
112 [associations](#)¹². For example, `x-scheme-handler/http` is used in desktop environ-
113 ments such as GNOME to designate an application as capable of acting as a
114 general-purpose web browser, and we will do the same here.

115 Services that parse desktop files should use [the implementation in GLib](#)¹³, or
116 an Apertis-specific API built on top of that implementation.

117 **Additional recommended keys** The following additional keys are defined
118 in the `[Desktop Entry]` group.

- 119 • `X-GNOME-FullName` (`localestring`): The human-readable full name of the ap-
120 plication, such as `Test Web Browser`. This key is already used by the GLib
121 library, and by desktop environments based on it (such as GNOME). Like
122 `Name`, this is a “`localestring`”: non-English versions can be provided with
123 syntax like `X-GNOME-FullName[pt_BR]=Navegador Internet de teste`.

124 Simple applications (one entry point)

125 This is the simple case where an entry point has one “view”, for example a web
126 browser.

```
127 # cat org.test.Web.desktop
128 [Desktop Entry]
129 Type=Application
130 Name=Test Browser
131 GenericName=Browser
132 X-GNOME-FullName=The Test Web Browser
133 Exec=test-browser %U
134 Categories=Network;WebBrowser;
135 MimeType=text/html;x-scheme-handler/http;x-scheme-handler/https;
```

¹⁰<https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#localized-keys>

¹¹https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_framework/interface_discovery/

¹²https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_framework/content_hand-over/

¹³<https://gitlab.gnome.org/GNOME/glib/-/blob/main/gio/gdesktopappinfo.c#L2007>

136 Icon=applications-internet

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

138 Some bundles might have an entry point that exists only to be started as a
139 side-effect of other operations, for instance to [handle URIs and content-types](#)¹⁴.
140 Those entry points would have `NoDisplay=true` to hide them from the menus;
141 that is the only difference.

142 **Multiple-view applications**

143 Some bundles have more than one entry in the system menus; the example
144 referred to previously would be an audio player with entry points for Artists,
145 Songs, and Albums. We propose to represent these with one `.desktop` file per
146 menu entry.

147 In this model, each menu entry is a `.desktop` file. The audio player
148 would install `org.test.Audio.Artists.desktop`, `org.test.Audio.Songs.desktop` and
149 `org.test.Audio.Albums.desktop`. In addition, it would install `org.test.Audio.desktop`
150 with `NoDisplay=true`.

151 The running instance of the application would always identify itself as
152 `org.test.Audio`, and the other three `.desktop` files would be linked to it by way
153 of being in the same app bundle.

154 When using [D-Bus activation](#)¹⁵ for applications (which is recommended), the
155 application would have separate D-Bus `.service` files for all four names, would
156 take all four bus names and their corresponding object paths at runtime, and
157 would export the `org.freedesktop.Application` API at all four paths; but all
158 of them would have `SystemdService=org.test.Audio.service` to ensure that only
159 one activation occurs. The `Activate`, `Open` or `ActivateAction` method on each bus
160 name would open the relevant view.

161 The result would look something like this:

```
162 # org.test.Audio.desktop
163 [Desktop Entry]
164 Type=Application
165 Name=Frampton
166 GenericName=Audio Player
167 X-GNOME-FullName=The Test Audio Player
168 Exec=test-audio %F
169 Categories=Audio;Player;Music;
170 MimeType=audio/mpeg;
171 NoDisplay=true;
```

¹⁴https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_framework/content_hand-over/

¹⁵<http://standards.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#dbus>

```
172 Icon=music

173 # org.test.Audio.Artists.desktop
174 [Desktop Entry]
175 Type=Application
176 Name=Frampton — Artists
177 GenericName=Artists
178 Exec=test-audio --artists
179 Categories=Audio;Player;Music;
180 Icon=music-artist

181 # org.test.Audio.Albums.desktop
182 [Desktop Entry]
183 Type=Application
184 Name=Frampton — Albums
185 GenericName=Albums
186 Exec=test-audio --albums
187 Categories=Audio;Player;Music;
188 Icon=music-album

189 # org.test.Audio.Songs.desktop
190 [Desktop Entry]
191 Type=Application
192 Name=Audio — Songs
193 GenericName=Songs
194 Exec=test-audio --songs
195 Categories=Audio;Player;Music;
196 Icon=music-track
```