Application entry points

Contents

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 listed in the "device simulator".

[1] https://www.apertis.org/glossary/#application-bundles
[2] https://www.apertis.org/glossary/#graphical-program
[3] https://www.apertis.org/concepts/archive/application_framework/content_hand-over/

Security and privacy considerations

The list of installed store application bundles is considered to be private information, for several reasons:

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

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

- Certain platform components such as the app launcher require the ability to list store application bundles and/or their entry points. They must be able to do so.
- Store applications with special permissions might also be allowed to list store application bundles and/or their entry points.
- Store applications may list the entry points that advertise a particular *public interface*, as defined in the Interface discovery[4] design.
- Store applications without special permissions must not be able to enumerate store application bundles that do not contain an entry point advertising a public interface, either directly or by enumerating entry points and inferring the existence of a bundle from its entry points.

Menu entries

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

Background Services

Background services in the current Flatpak distribution model do *not* use entry points; rather, they request the ability to run arbitrary command lines in the background at runtime via the XDG Background portal[5].

[4] https://www.apertis.org/concepts/archive/application_framework/interface_discovery/

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

Non-requirements

System services[6] are outside the scope of this design.

Recommendation

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

App identification

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

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

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

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

All names should be allocated in a namespace controlled by the author of the bundle —in particular, Apertis applications should be in `org.apertis`. Sample code that is not intended to be used in production should be placed in `com.example`, with `org.example` and `net.example` also available for code samples that need to demonstrate the interaction between multiple namespaces (we prefer `com.example`, as a hint to developers that reversed domain names do not always start with "org").

Desktop entries

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

The localestring[10] mechanism is used for translated strings.

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

[6] https://www.apertis.org/glossary/#system-service
[7] application-bundle-metadata.md
[8] https://en.wikipedia.org/wiki/Reverse_domain_name_notation
[9] http://standards.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html
[10] https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#localized-keys

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

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

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

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

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

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

Simple applications (one entry point)

This is the simple case where an entry point has one "view", for example a web browser.

```
# cat org.test.Web.desktop
[Desktop Entry]
Type=Application
Name=Test Browser
GenericName=Browser
X-GNOME-FullName=The Test Web Browser
Exec=test-browser %U
Categories=Network;WebBrowser;
MimeType=text/html;x-scheme-handler/http;x-scheme-handler/https;
Icon=applications-internet
```

Entry points which do not appear in the menus

Some bundles might have an entry point that exists only to be started as a side-effect of other operations, for instance to handle URIs and content-types[14].

[11] https://www.apertis.org/concepts/archive/application_framework/interface_discovery/
[12] https://www.apertis.org/concepts/archive/application_framework/content_hand-over/
[13] https://gitlab.gnome.org/GNOME/glib/-/blob/main/gio/gdesktopappinfo.c#L2007
[14] https://www.apertis.org/concepts/archive/application_framework/content_hand-over/

Those entry points would have `NoDisplay=true` to hide them from the menus; that is the only difference.

Multiple-view applications

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

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

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

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

The result would look something like this:

```
# org.test.Audio.desktop
[Desktop Entry]
Type=Application
Name=Frampton
GenericName=Audio Player
X-GNOME-FullName=The Test Audio Player
Exec=test-audio %F
Categories=Audio;Player;Music;
MimeType=audio/mpeg;
NoDisplay=true;
Icon=music

# org.test.Audio.Artists.desktop
[Desktop Entry]
Type=Application
Name=Frampton — Artists
GenericName=Artists
Exec=test-audio --artists
```

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

```
Categories=Audio;Player;Music;
Icon=music-artist

# org.test.Audio.Albums.desktop
[Desktop Entry]
Type=Application
Name=Frampton — Albums
GenericName=Albums
Exec=test-audio --albums
Categories=Audio;Player;Music;
Icon=music-album

# org.test.Audio.Songs.desktop
[Desktop Entry]
Type=Application
Name=Audio — Songs
GenericName=Songs
Exec=test-audio --songs
Categories=Audio;Player;Music;
Icon=music-track
```