Apertis Application Bundle Specification

Contents

Version: 1.2.0

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119[1].

This specification uses semantic versioning[2]. After version 1.0.0 is finalized, the major version number (first component) will be incremented if a change

[1] https://tools.ietf.org/html/rfc2119

[2] http://semver.org/

makes previously-valid application bundles cease to work or be valid, for example changing “MAY” to “MUST” or “MUST NOT”. The minor version number (second component) will be incremented if a change makes previously-invalid application bundles valid, for example changing “MUST” or “MUST NOT” to “MAY”. The micro version number (third component) will be incremented for editorial changes that do not affect the validity of an application bundle.

Introduction

This document aims to provide a stable filesystem layout for Apertis store application bundles[3] that can remain valid across multiple versions.

To keep older application bundles installable on newer Apertis releases, we anticipate that incompatible changes (incrementing the major version) are to be made very infrequently. If necessary, Apertis framework components might be made to support multiple major versions of this specification.

A secondary goal of this specification is to provide a basis for the structure of built-in application bundles[4]. Authors of built-in application bundles do not necessarily need to limit themselves to the baseline set by this specification, since a built-in application bundle will only be upgraded or rolled back at the same time as a corresponding upgrade or rollback for the Apertis platform. However, by following the requirements and recommendations in this specification, a built-in application bundle author can minimize the changes necessary between Apertis platform releases. Please see Appendix: built in application bundles[5] for differences between the required structure of store bundles and the required structure of built-in application bundles.

App-bundles contain some or all of the following files:

- `bin/*` (Executables)
- `etc/apparmor.d/Applications.`*bundle-ID* (AppArmor profile)
- `lib/*` (Libraries)
- `libexec/*` (Executables)
- `share/applications/`*entry-point*`.desktop` (Entry points)
- `share/glib-2.0/schemas/`*schema-ID*`.gschema.xml` (GSettings schemas)
- `share/glib-2.0/schemas/gschemas.compiled` (GSettings schemas)
- `share/icons/hicolor/64x64/apps/`*bundle-ID*`.png` (Icon for the bundle)
- `share/icons/hicolor/64x64/apps/`*entry-point*`.png` (Icons for entry points)
- `share/icons/`*theme*`/` (Icons for use by the bundle)
- `share/locale/*/LC_MESSAGES/*.mo` (Localized strings)
- `share/metainfo/`*bundle-ID*`.appdata.xml` (Bundle metadata)
- `share/metainfo/`*bundle-ID*`.metainfo.xml` (Bundle metadata)

[3] https://sjoerd.pages.apertis.org/apertis-website/concepts/applications/#software-categories

[4] https://sjoerd.pages.apertis.org/apertis-website/concepts/applications/#software-categories

[5]

- `share/themes/`*theme*`/*` (Theme data for use by the bundle)
- `share/*` (Generic resource data)

all of which will be installed relative to `/Applications/`*bundle-ID*.

To reduce its length, this specification does not generally provide rationale for its requirements. Please see the Apertis concept designs[6] for design rationale, in particular the Applications concept design[7], Application Layout concept design[8] and Application Entry Points concept design[9].

Bundle ID

Each Apertis application bundle has a *bundle ID*, which MUST consist of two or more components separated by dots (U+002E FULL STOP). Each component MUST start with an ASCII letter (`A-Z`, `a-z`) or underscore `_`, and contain only ASCII letters, underscores and ASCII decimal digits (`0-9`). Bundle IDs MUST NOT contain non-ASCII characters, for example accented letters such as `ä`.

> Note that these are the same as the requirements for a D-Bus interface name[10], which are stricter than the requirements for a D-Bus bus name or a GApplication application ID[11]: every bundle ID is a valid bus name and a valid GApplication application ID, but not every bus name or application ID is a valid bundle ID.

Reversed domain name

The author of an application MUST choose a bundle ID that starts with a reversed domain name[12] controlled by that author, with any hyphen/minus signs `-` replaced by underscores `_`, and `_` prepended to any component that starts with a digit.

For example, the owner of the domain name `collabora.com` controls the reversed domain name `com.collabora` and might choose to name an app bundle `com.collabora.ShoppingList`.

Domain names with hyphen/minus signs, or with components starting with a digit, require special treatment to avoid syntactically invalid bundle IDs. If the owner of `7-zip.org` wishes to base bundle IDs on that domain name, they MUST use a bundle ID starting with `org._7_zip`; for example, they might choose to name an app bundle `org._7_zip.Archiver`.

[6] https://designs.apertis.org/
[7] https://sjoerd.pages.apertis.org/apertis-website/concepts/applications/
[8] https://sjoerd.pages.apertis.org/apertis-website/concepts/application-layout/
[9] https://sjoerd.pages.apertis.org/apertis-website/concepts/application-entry-points/
[10] https://dbus.freedesktop.org/doc/dbus-specification.html#message-protocol-names
[11] https://developer.gnome.org/gio/stable/GApplication.html#g-application-id-is-valid
[12] https://en.wikipedia.org/wiki/Reverse_domain_name_notation

Top-level directory

Each application bundle is made available on the user's system as a subdirectory of `/Applications` whose name is the same as the bundle ID. App bundles MUST NOT include any file outside that directory.

For example, the app bundle `com.example.ShoppingList` would use a top level directory `/Applications/com.example.ShoppingList/`.

For brevity, this document will refer to this directory as `${prefix}`.

Bundle metadata

Each app-bundle MUST install exactly one file in the `${prefix}/share/metainfo/` directory. The contents of that file are interpreted according to the AppStream upstream XML[13] specification.

This table provides a summary of the relevant tags. All other tags are either not recommended for any type of bundle, or not allowed.

Tag	Status
`id`	required, must be bundle ID
`name`	required
`summary`	recommended
`description`	recommended
`developer_name`	recommended
`metadata_license`	required, should be `CC0-1.0`
`project_license`	optional
`url`	optional
`releases`	required
`releases` → `release`	required, exactly one
`provides`	optional
`provides` → `dbus`	optional
`provides` → *any other*	not allowed
`custom` → `value`	optional

If the app-bundle has Entry points, the file MUST be named either `${bundle_id}.appdata.xml` or `${bundle_id}.metainfo.xml`, replacing `${bundle_id}` with the Bundle ID. In this case the `component` tag MUST have its `type` attribute set to `desktop`.

If the app-bundle does not have Entry points, the file MUST be named `${bundle_id}.metainfo.xml`, again replacing `${bundle_id}` with the Bundle ID. In this case the `component` tag MUST NOT have a `type` attribute.

The `id` tag MUST contain exactly the Apertis Bundle ID.

[13] https://www.freedesktop.org/software/appstream/docs/chap-Metadata.html

The `name` tag MUST contain a human-readable name for the app-bundle, for example `Shopping List`.

The `summary`, `description` and `developer_name` tags SHOULD be present, with the contents described by the AppStream upstream XML[14] specification.

The `metadata_license` tag MUST be present, and MUST contain the identifier of a permissive license under which the metadata can be redistributed. This license SHOULD be the Creative Commons Zero license[15], `CC0-1.0`, allowing unlimited redistribution of the metadata with or without modifications (for example in the user interface of an app-store).

The `metadata_license` does not imply anything about the terms under which the app-bundle itself can be distributed: app-bundles themselves MAY be distributed under any license of their copyright holder's choice, including proprietary licenses. The bundle metadata MAY represent that license in the `project_license` tag, as described in the AppStream upstream XML[16] specification.

The `url` tag MAY be present, with the types and contents described by the AppStream upstream XML[17] specification.

The `releases` tag MUST be present, and MUST contain exactly one `release` tag. The `release` tag MUST have a `version` attribute. Its value MUST start with a digit and contain only digits and U+002E FULL STOP characters. Note that this is a more strict requirement than in the AppStream upstream XML specification, which allows additional `release` tags describing older releases.

> Future directions:
>
> This is a very strict versioning syntax, matching what Ribchester accepts in Apertis 16.09. We should consider expanding this in a future minor version of this specification to be able to accept dpkg-style versions like `1.2.3~beta1+bugfix2`. This will require a formal specification for how these version numbers are to be compared, possibly deferring to Debian Policy[18].
>
> The Applications concept design[19] calls for the version number to be split into an *application version* and a *store version*, analogous to the roles of the *upstream version* and *Debian revision* in Debian.
>
> **Open question**: is the store version encoded in the `release` tag, or is it stored in a `custom` tag or separately?

[14] https://www.freedesktop.org/software/appstream/docs/chap-Metadata.html
[15] https://creativecommons.org/publicdomain/zero/1.0/
[16] https://www.freedesktop.org/software/appstream/docs/chap-Metadata.html
[17] https://www.freedesktop.org/software/appstream/docs/chap-Metadata.html
[18] https://www.debian.org/doc/debian-policy/ch-controlfields.html#s-f-Version
[19] https://sjoerd.pages.apertis.org/apertis-website/concepts/applications/

The `provides` tag MAY be present. It MAY contain a `dbus` tag, with its `type` attribute set to `user`, for each well-known name provided by an entry point in this application bundle. It MUST NOT contain any of the other child tags that can be provided.

The `mimetypes` tag MUST NOT be present. In Apertis, content-type support is handled by Entry points.

The `project_group` tag MUST NOT be present.

Tags that are not specified in the AppStream upstream XML specification MUST NOT be present, with the exception of `custom` (see Extended bundle metadata, below).

Tags not specified in this document, in particular `screenshots`, `suggests`, `translation` and `update_contact`, SHOULD NOT be present.

Extended bundle metadata

The bundle metadata MAY include one `custom` tag at the next level of hierarchy below `component`. This tag MAY contain `value` child tags, each with a `key` attribute and XML character data (text) content. It MUST NOT contain other child tags or text.

Later versions of this specification will define keys starting with `X-Apertis-`. Keys with that prefix that are not defined in this document MUST NOT be present. The current version of this document does not define any such keys.

Other vendors MAY define keys starting with `X-` followed by a name distinctive to the vendor.

Other keys SHOULD NOT be present.

For example, if a future version of this specification defined a key `X-Apertis-ExampleColour`, and a vendor Wayne Industries defined a key `X-Wayne-BatmobileCompatible`, this might result in bundle metadata like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<component type="desktop">
  <id>net.example.Extended</id>
  <custom>
    <value key="X-Apertis-ExampleColour">#00cc00</value>
    <value key="X-WayneIndustries-BatmobileCompatible">true</value>
  </custom>
  ··· additional metadata here ···
</component>
```

> It is anticipated that this mechanism will be used for Apertis-specific or automotive-specific extensions that are considered insufficiently general to be included in the AppStream standard.

AppArmor profile

Apertis uses AppArmor[20] to provide security between application bundles[21]. Each app-bundle MUST install exactly one AppArmor profile file at `${prefix}/etc/apparmor.d/Applications.${bundle_id}`, replacing `${bundle_id}` with the Bundle ID.

This file MUST define exactly one AppArmor profile. Its name MUST be exactly `/Applications/${bundle_id}/**`, again replacing `${bundle_id}` with the bundle ID. It MUST NOT have any local profiles[22] (also known as child profiles or subprofiles), and in particular MUST NOT have any hats[23] (which are a special case of local profiles).

This file SHOULD contain the following rules, replacing `@BUNDLE_ID@` with the Bundle ID throughout:

[20] https://gitlab.com/apparmor/apparmor/wikis/About

[21] https://sjoerd.pages.apertis.org/apertis-website/designs/security/#security-between-applications

[22] https://gitlab.com/apparmor/apparmor/wikis/AppArmor_Core_Policy_Reference#local-profiles-and-hats

[23] https://gitlab.com/apparmor/apparmor/wikis/AppArmor_Core_Policy_Reference#local-profiles-and-hats

```
/Applications/@BUNDLE_ID@/** {
  #include <abstractions/chaiwala-base>
  #include <abstractions/dbus-session-strict>
  #include <abstractions/fonts>

  /Applications/@BUNDLE_ID@/{bin,libexec}/* pix,
  /Applications/@BUNDLE_ID@/{bin,lib,libexec}/{,**} mr,
  /Applications/@BUNDLE_ID@/share/{,**} r,

  owner /var/Applications/@BUNDLE_ID@/users/** rwk,

  owner link
        subset /var/Applications/@BUNDLE_ID@/users/**
            -> /var/Applications/@BUNDLE_ID@/users/**,

  dbus send
    bus=session
    path=/org/freedesktop/DBus
    interface=org.freedesktop.DBus
    member={RequestName,ReleaseName}
    peer=(name=org.freedesktop.DBus),
  dbus bind bus=session name="@BUNDLE_ID@",
  dbus bind bus=session name="@BUNDLE_ID@.*",
 dbus (send, receive) bus=session peer=(label=/Applications/@BUNDLE_ID@/**),
  dbus receive bus=session peer=(label=/usr/bin/canterbury),

  signal receive peer=/usr/bin/canterbury,
}
```

The profile MAY add additional permissions. The app-store curator is expected to check additional permissions carefully.

> Future direction: the profile should be generated from simpler metadata[24] in a future minor version of this specification.

> Future direction: if we recommend particular interpreters — for example /bin/sh for wrappers that set environment variables, a JavaScript or Python interpreter for interpreted app code, or a webapp runtime for HTML5 apps — then the generic profile recommendation should allow those interpreters to be used.

[24] https://phabricator.apertis.org/T311

Entry points

Each app-bundle MAY install `.desktop` files in the `${prefix}/share/applications/` directory. The contents of that file are interpreted according to the Desktop Entry Specification[25].

App bundles are not required to install any entry points at all, but many features can only be provided by an app bundle that has entry points: Graphical programs in the main menu MUST have an entry point, and Content type and uri scheme handlers MUST have a Main entry point.

The name of each desktop entry file, excluding the `.desktop` extension, is called the Entry point ID.

This table provides a summary of the allowed, recommended and optional fields. All other fields are either not recommended for any type of entry point, or not allowed. A table cell containing a literal value indicates that the field is required and must have exactly that value.

Field	Main entry point	Other graphical programs	Agents
`Categories`	required	required	not recommended
`Exec`	required	required	required
`GenericName`	optional	optional	optional
`Icon`	required	required	not recommended
`Interfaces`	optional	optional	optional
`MimeType`	optional	not allowed	not allowed
`Name`	recommended	recommended	recommended
`NoDisplay`	optional	optional	`true`
`OnlyShowIn`	`Apertis;`	`Apertis;`	`Apertis;`
`Path`	optional	optional	optional
`Type`	`Application`	`Application`	`Application`
`X-Apertis-CategoryIcon`	required	required	not recommended
`X-Apertis-CategoryLabel`	required	required	not recommended
`X-Apertis-Type`	`application`	`application`	`agent-service`
`X-GNOME-FullName`	optional	optional	optional
`DBusActivatable`	`true` recommended	`true` recommended	`true` recommended
`X-Apertis-ServiceExec`	recommended	optional	not allowed
`X-Apertis-ParentEntry`	not recommended	optional	not allowed

General fields for all entry points

`Type` MUST be set to `Application`.

`OnlyShowIn` MUST be set to `Apertis;`.

[25] http://standards.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html

`Exec` MUST be present. The first word in `Exec` MUST be the absolute path to an executable in either `${prefix}/bin` or `${prefix}/libexec`.

Subsequent words in `Exec` MUST NOT use the `%` placeholders such as `%F`.

> The Canterbury application manager does not support those placeholders.

Subsequent words in `Exec` MUST NOT be exactly `app-name`, `play-mode` or `url`, and SHOULD NOT be exactly `menu-entry`.

> These words cause unexpected special behaviour in Apertis 16.06. After this special behaviour has been removed, future minor versions of this specification should remove this limitation.

`Name` SHOULD be specified. Its value MAY be a "brand name" such as `Firefox`, a generic name such as `Web browser`, or a combination of the two such as `Firefox web browser`.

`GenericName` SHOULD be specified if its value would differ from `Name`. If present, its value MUST be a generic (unbranded) name such as `Web browser`, for use in user interfaces whose designer wishes to standardize on generic names.

`X-GNOME-FullName` SHOULD be specified if its value would differ from `Name`. If present, its value SHOULD be a full name incorporating both the brand name and the generic name, for example `Firefox web browser`, suitable for use in situations where it is necessary to disambiguate between entry points with the same `GenericName` (for example if both Firefox and Chrome have `GenericName=Web browser`).

Translated versions of these names, such as `Name[fr]`, MAY be present using the localestring[26] mechanism defined in the Desktop Entry Specification.

`Path` MAY be set, with its usual meaning (it sets the current working directory for the program). If `Path` is not set, programs in the app-bundle will inherit the working directory of the parent process, and MUST NOT assume that it will take any particular value.

`Interfaces` MAY be set, for interface discovery[27].

`DBusActivatable` MAY be present and set to `true`, as described in D-Bus activation.

`X-Apertis-ServiceExec` MAY be set, as described in D-Bus activation.

`X-Apertis-ParentEntry` MAY be set, as described in Multiple views.

The following keys MUST NOT be present:

- `Encoding` (the encoding MUST be UTF-8, which is the default)

[26] https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#localized-keys

[27] https://sjoerd.pages.apertis.org/apertis-website/concepts/interface_discovery/

- `Hidden` (this misleadingly named key is used to mark entry points as deleted, which is not useful in this context)
- `NotShowIn`
- `StartupNotify`
- `StartupWMClass`
- `Terminal`
- `URL`
- `Version` (version 1.0 of the Desktop Entry specification is assumed)

The following keys SHOULD NOT be present, and application bundles SHOULD NOT rely on their normal functionality (if any):

- `Actions`
- `Comment`
- `Environment`
- `Keywords`
- `TryExec`
- `X-Apertis-AudioChannelName`
- `X-Apertis-AudioResourceOwner`
- `X-Apertis-AudioRole`
- `X-Apertis-BackgroundState`
- `X-Apertis-BandwidthPriority`
- `X-Apertis-DataExchangeRules` (obsolete)
- `X-Apertis-ManifestUrl` (obsolete)
- `X-Apertis-SettingsIcon` (set an `Icon` instead)
- `X-Apertis-SettingsName` (set the `name` in the Bundle metadata instead)
- `X-Apertis-SettingsPath` (use the mechanism described in GSettings schemas instead)
- `X-Apertis-SplashScreen`
- `X-Apertis-WindowName` (obsolete)

> Future directions:
>
> `X-Apertis-AudioRole`, `X-Apertis-BackgroundState` and `X-Apertis-BandwidthPriority` are under consideration for a future minor version of this specification, but are not currently considered to be stable.

Entry point ID

Each entry point MUST have an *entry point ID*, which is a string with the same syntax requirements as a Bundle ID. The name of the `.desktop` file MUST be the entry point ID followed by `.desktop`.

Like the Bundle ID, all entry point IDs in an app-bundle MUST start with a Reversed domain name controlled by the author. It is RECOMMENDED that all entry point IDs in an app-bundle either match its bundle ID exactly, or start with the bundle ID followed by a dot.

A single executable program MAY be represented by more than one entry point.

If a program will request a D-Bus well-known bus name to provide interfaces to graphical programs in the same bundle, the well-known bus name MUST be the same as one of its entry point IDs.

Main entry point

Each app-bundle that has any entry points SHOULD have an entry point whose Entry point ID is exactly the Bundle ID. This entry point is referred to as the *main entry point*, and MUST be a [graphical program][Graphical programs].

Certain metadata fields of the main entry point, including its `Categories`, `Icon` and `MimeType`, are copied into the cache of bundle metadata during installation and hence made available to platform applications.

Content type and URI scheme handlers

The Main entry point MAY be registered as the content type handler[28] for media types such as `audio/mpeg`, by setting `MimeType` to a list of content types[29], each followed by a semicolon `;`. Non-main entry points MUST NOT be content type handlers. For example, a media player with support for the MP3 and RealAudio formats might use `MimeType=audio/mpeg;audio/vnd.rn-realaudio;` in its entry point.

The Main entry point MAY be registered as the handler for URI schemes[30] such as `tel` or `http`, by including an entry in `MimeType` for the pseudo-content-type `x-scheme-handler/`*scheme*, for example `x-scheme-handler/http` for a web browser. Non-main entry points MUST NOT be URI scheme handlers.

If the entry point implements D-Bus activation, sending the `org.freedesktop.Application.Open` method call[31] to the object path corresponding to its entry point ID MUST result in it attempting to open the URI or URIs passed as parameters.

Graphical programs

Each graphical program (user interface, HMI) that will be directly launched by the user MUST have an entry point. Each graphical program that will be associated with content types or URIs MUST have an entry point.

A graphical program MAY have more than one entry point, to appear in menus more than once (for example, the Frampton media player uses this[32] to appear three times under the names Artists, Albums and Songs).

Graphical programs MUST set `X-Apertis-Type` to `application`.

[28] https://sjoerd.pages.apertis.org/apertis-website/concepts/content_hand-over/

[29] https://www.iana.org/assignments/media-types/media-types.xhtml

[30] https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml

[31] https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#dbus

[32] https://gitlab.apertis.org/appfw/frampton/tree/v0.6.1/scripts

> This is required by Apertis 16.09, but might be phased out in a later minor version of this specification.

If a graphical program is intended to be shown in the menus, `NoDisplay` MUST NOT be specified. Otherwise, it MUST be specified and set to `true` (for example, the Frampton media player uses this to allow its *main entry point* to be associated with media file types while hiding it from the menus).

`Categories` MUST be set to a list of appropriate menu categories from the freedesktop.org Desktop Menu Specification[33], each followed by a semicolon `;`. There MUST be at least one Main Category.

`X-Apertis-CategoryLabel` MUST be set to the human-readable English name of a single category, which MUST be in title-case with no special formatting (for example, `Video & TV` is correct, while `V I D E O  &  T V` is not).

> This is required by Apertis 16.09, but should be phased out in favour of having launchers parse Categories in a later minor version of this specification.

`X-Apertis-CategoryIcon` MUST be set to the name of the icon to be used for the category in launchers, with no `/` characters or file-type extension, for example `icon_music_AC`. The icon MUST be chosen from among those provided by the platform's launcher (the allowed values are therefore platform-specific).

> This is required by Apertis 16.09, but should be phased out in favour of having launchers parse Categories in a later minor version of this specification.

`Icon` MUST be set to the name of either the Icon for the bundle or one of the Icons for entry points, as a bare icon name (without any `/` characters, and without a file-type extension such as `.png` or `.svg`). In particular, this implies that its string value MUST match either the Bundle ID, or the Entry point ID of an entry point.

Multiple views

Some application designs have a group of entry points that are all implemented by invoking the same executable with different parameters, all implemented in the same process. For example, a music player might have separate entry points to view the music library grouped by artist or album, or as a single flat list of songs.

In applications that work like this, one of these views MUST be nominated to be the *parent entry point*, with the others as *child entry points*. The parent will usually be the Main entry point, although this is not required. The main entry point SHOULD NOT be a child entry point.

[33] http://standards.freedesktop.org/menu-spec/latest/apa.html

A parent entry point MUST NOT have the `X-Apertis-ParentEntry` field. It MUST set `DBusActivatable` to `true`, and implement [D-Bus activation] for its own entry point ID and the entry point IDs of all associated child entry points. It SHOULD set `X-Apertis-ServiceExec`.

Child entry points MUST set the `X-Apertis-ParentEntry` field to the Entry point ID of the parent entry point, and MUST set `DBusActivatable` to `true`. They MUST NOT set `X-Apertis-ServiceExec`.

Agents and other non-graphical programs MUST NOT be parent or child entry points.

D-Bus activation

Programs in an app-bundle MAY declare that they implement *D-Bus activation* by setting `DBusActivation` to `true` in each entry point that starts the same program.

Graphical program entry points that set `DBusActivation` to `true` and do not have an `X-Apertis-ParentEntry` field SHOULD also have an `X-Apertis-ServiceExec` field. The `X-Apertis-ServiceExec` field has the same syntax as the standard `Exec` field.

Agents MUST NOT have an `X-Apertis-ServiceExec` field, since their `Exec` field has essentially the same meaning.

We define the *service activation command line* to be the `X-Apertis-ServiceExec` field if present, or the `Exec` field otherwise.

The service activation command line MUST be a command-line that will start the program without opening any graphical windows, such that it will be ready to receive D-Bus requests. If a program uses the GApplication[34] API (which is recommended), then the service activation command line for graphical programs will typically be the absolute path of the executable followed by a space and the `--gapplication-service` argument, for example `X-Apertis-ServiceExec=/Applications/com.example.ShoppingList/bin/main --gapplication-service`, while the service activation command line for agents and other non-graphical programs (with the `G_APPLICATION_IS_SERVICE` flag[35]) will typically just be the path to the executable.

When the service activation command line for a graphical program is launched, the resulting process MUST export a D-Bus object path that is derived from the entry point ID by prepending `/` and replacing each `.` with `/`, then request a well-known name equal to the entry point ID. The interfaces of that object path MUST include at least the `org.freedesktop.Application` interface[36],

[34] https://developer.gnome.org/gio/stable/GApplication.html

[35] https://developer.gnome.org/gio/stable/GApplication.html#G-APPLICATION-IS-SERVICE:CAPS

[36] https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#dbus

and MAY include additional standard or non-standard interfaces such as the `org.gtk.Application` interface[37] used by GLib's GApplication[38] objects. When launched in this way, the process MUST NOT behave as though any of its entry points were activated until it receives an appropriate D-Bus method call; in particular, it MUST NOT open any windows until it is told to do so.

When the service activation command line for an agent or non-graphical program is launched, the resulting process MAY export a D-Bus object path implementing `org.freedesktop.Application` as above, but is not required to do so. If it does, the `Activate` and `Open` methods are not required to be implemented, since they are unlikely to be useful for non-graphical programs.

If an entry point has an `X-Apertis-ParentEntry` field (a [child entry point][Multiple views]), when the parent entry point named in that field is started by its service activation command line, the resulting process MUST also export D-Bus object paths and request well-known names corresponding to the entry point IDs of each of its child entry points.

For graphical programs, sending the `org.freedesktop.Application.Activate` D-Bus method call[39] to one of the object paths described above MUST result in the program displaying whatever window is appropriate for the corresponding entry point. If the graphical program implements [content-type handling][Content type and uri scheme handlers], then the same is true for the `org.freedesktop.Application.Open` method[40]. This requirement is not applicable to agents and other non-graphical programs.

The process MAY export additional object paths and interfaces. It SHOULD NOT request additional well-known names.

When the `Exec` command of a D-Bus-activatable graphical entry point is launched, the resulting process MUST arrange for a program to be run (directly or indirectly) that will request the well-known name corresponding to that entry point ID, export the corresponding D-Bus object path, and behave as though that object path had received an `Activate` or `Open` method call, modified according to the command-line arguments if appropriate: in other words, it has behaviour similar to the `Exec` command of a non-D-Bus-activatable graphical program. If the program uses the GApplication[41] API, this will normally be achieved by setting `Exec` to the absolute path of the executable, with no arguments, for example `Exec=/Applications/com.example.ShoppingList/bin/main`. This requirement is not applicable to agents and other non-graphical programs.

[37] https://wiki.gnome.org/Projects/GLib/GApplication/DBusAPI

[38] https://developer.gnome.org/gio/stable/GApplication.html

[39] https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#dbus

[40] https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#dbus

[41] https://developer.gnome.org/gio/stable/GApplication.html

See the specification of the Application interface[42] for more details about its methods.

Agents

Each agent (background service) MUST have an entry point.

Agents MUST set `X-Apertis-Type` to `agent-service`.

> This is required by Apertis 16.09, but might be phased out in a later minor version of this specification.

Agents MUST set `NoDisplay` to `true`.

Agents SHOULD set `DBusActivatable` to `true`, and implement D-Bus activation as described above.

Paths for other file types

Executables

Any executable programs in the app-bundle MUST be installed in either the `${prefix}/bin` or `${prefix}/libexec` directory, or a descendant directory in `${prefix}/libexec`. For example, these paths are valid:

```
${prefix}/bin/my-executable
${prefix}/libexec/my-helper-executable
${prefix}/libexec/other-helper/other-helper-executable
```

> Suitable directories are conveniently available as `${bindir}`, `${libexecdir}` and ${pkglibexecdir}‘ when using Automake.

Libraries

An Apertis application bundle MAY contain private libraries for use by that application bundle, for example shared libraries written in C or C++, or Python modules.

If present, architecture-dependent library files MUST be located in the `${prefix}/lib` directory or a descendant of that directory. Architecture-independent library files such as “pure Python” modules MUST be located in either the `${prefix}/lib` or `${prefix}/share` directory, or a descendant of one of those directories.

For example, the app bundle `com.example.ShoppingList` might contain library

[42] https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#dbus

files `/Applications/com.example.ShoppingList/lib/libwebapi.so.0` or `/Applications/com.example.ShoppingList/lib/python3/webapi/__init__.py`.

Native executables SHOULD be linked with a `DT_RPATH` pointing to the location of their required libraries. For example, the ShoppingList app bundle described above might be linked using `gcc -Wl,-rpath=/Applications/com.example.ShoppingList/lib`.

> If the app bundle is built using GNU automake and libtool, this will typically be done automatically.

Programs in app-bundles MUST NOT assume that any special environment variables to locate libraries, such as `LD_LIBRARY_PATH`, `GI_TYPELIB_PATH` or `PYTHONPATH`, will be set by the application framework. For example, if the ShoppingList app bundle described above needs to be able to load `/Applications/com.example.ShoppingList/lib/python3/webapi/__init__.py` via the Python statement `import webapi`, it cannot assume that `/Applications/com.example.ShoppingList/lib/python3` is already in `sys.path`. Its main executable might prepend that directory to `sys.path`, or its main executable might be a shell script that sets `PYTHONPATH` and then runs the underlying Python code with `exec`.

> A possible change in future minor versions of this specification would be to set a specified list of environment variables used by a specified set of recommended libraries, such as `LD_LIBRARY_PATH` for libc and `GI_TYPELIB_PATH` for GObject-Introspection. Python is not among our recommended frameworks, so we would probably still not include `PYTHONPATH`.

For each native ELF library, the app-bundle MUST contain a file whose name exactly matches the SONAME[43] (ELF `DT_SONAME`) of the library, in a directory that will be searched by all executables that use that library (for example via `DT_RPATH` or `LD_LIBRARY_PATH`). This file MUST either be a regular file (the library itself), or a symbolic link to the library's "real name".

Building and installing shared libraries using GNU libtool is RECOMMENDED: libraries built like this will typically have a correct symbolic link from the `SONAME` to the "real name" without further action from the developer.

For example, if the ShoppingList app-bundle has executables linked to a private library whose `SONAME` is `libwebapi.so.0`, it might include a regular file with exactly that name; or it might include a regular file named `libwebapi.so.0.1.2`, and a symbolic link `libwebapi.so.0 → libwebapi.so.0.1.2`.

Icon for the bundle

The app-bundle MAY have an icon to represent the bundle as a whole, in a generic user interface icon theme. The generic user interface icon theme is rep-

[43] http://tldp.org/HOWTO/Program-Library-HOWTO/shared-libraries.html

resented by the reserved theme name `hicolor`, as required by the freedesktop.org Icon Theme Specification[44].

If the app-bundle has this icon, it MUST be in Portable Network Graphics[45] format, 64×64 pixels in size, and MUST be located at

```
${prefix}/share/icons/hicolor/64x64/apps/${name}.png
```

where `${name}` is set to the Bundle ID.

> **Open question**: I'm arbitrarily choosing 64x64 because that's what the AppStream specification uses, but do we have a different preferred size in Apertis?

To minimize display artefacts caused by resizing, the app-bundle MAY make this icon available in some or all of the additional sizes used in the freedesktop.org reference implementation[46] of the `hicolor` fallback theme (8, 16, 22, 24, 32, 36, 42, 48, 64, 72, 96, 128, 192, 256 or 512 pixels). These MUST be installed to the corresponding path with `64x64` replaced by the appropriate size.

The app-bundle MAY have variations of this icon that fit better in specific user interface themes. If present, these MUST be installed to the corresponding path with `hicolor` replaced by the name of the intended theme. For example, if a theme named `net.example.Metallic` is popular, an app-bundle might include a version of its own icon that has been designed to coordinate well with the Metallic theme, at

```
${prefix}/share/icons/net.example.Metallic/64x64/apps/${name}.png
```

Icons for entry points

Any entry point MAY have an icon to represent it. If present, it MUST be named in the same way as the icon for the bundle as a whole, except that `${name}` is set to the Entry point ID instead of the bundle ID.

Note that this means the Main entry point of the app-bundle will always use the same icon as the bundle itself.

[44] https://specifications.freedesktop.org/icon-theme-spec/icon-theme-spec-latest.html
[45] https://tools.ietf.org/html/rfc2083
[46] https://www.freedesktop.org/software/icon-theme/releases/

Icons for use by the bundle

The app-bundle MAY contain other icons. They SHOULD be arranged according to the freedesktop.org Icon Theme Specification[47].

For example, if the app-bundle is an email client, it might include a `mail-mark-important` icon for use by a “Mark as Important” button. If it has a generic version for use by unrecognised themes, and that generic version is 24 pixels in size, that version might be installed in:

```
${prefix}/share/icons/hicolor/24x24/actions/mail-mark-important.png
```

If the app-bundle also has a version for use by a popular theme named `net.example.Metallic`, it might install that as:

```
${prefix}/share/icons/net.example.Metallic/24x24/actions/mail-mark-important.png
```

The app-bundle MAY assume that it will be launched with the `XDG_DATA_DIRS` environment variable[48] set to a value that includes `${prefix}/share`, so that common icon theme implementations such as GtkIconTheme[49] will automatically use icons from the `${prefix}`.

Theme data for use by the bundle

The app-bundle MAY install theme data into subdirectories of `${prefix}/share/themes` whose names correspond to theme names.

The app-bundle MAY assume that it will be launched with the `XDG_DATA_DIRS` environment variable[50] set to a value that includes `${prefix}/share`, so that common theme implementations such as GtkCssProvider[51] will automatically use theme data from the `${prefix}`.

GSettings schemas

GSettings schemas[52] are used for preferences[53].

[47] https://specifications.freedesktop.org/icon-theme-spec/icon-theme-spec-latest.html

[48] https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html

[49] https://developer.gnome.org/gtk3/stable/GtkIconTheme.html

[50] https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html

[51] https://developer.gnome.org/gtk3/stable/GtkCssProvider.html

[52] https://developer.gnome.org/gio/stable/GSettings.html

[53] https://sjoerd.pages.apertis.org/apertis-website/designs/preferences-and-persistence/#preferences-approach

The app-bundle MAY install one or more GSettings schemas[54] into `${prefix}/share/glib-2.0/schemas/`. The filenames used MUST be the schema ID followed by `.gschema.xml`, optionally accompanied by enum definitions in a file named by the schema ID followed by `.enums.xml`. Each schema ID SHOULD either match the Bundle ID exactly, or start with the bundle ID followed by a dot.

If the app-bundle installs any schemas, then it MUST also install a compiled binary form of those schemas, in `${prefix}/share/glib-2.0/schemas/schemas.compiled`. The `glib-compile-schemas` tool can be used to compile this binary form.

The app-bundle MAY install a schema whose schema ID matches the Bundle ID exactly. If it does, then that schema's child schemas MUST all start with the bundle ID followed by a dot, and that schema and its child schemas will be made available in the system settings user interface.

If the app-bundle author does not intend for it to appear in the system settings user interface, then the app-bundle MUST NOT use its bundle ID as a schema ID. It MAY use an alternative schema ID such as `${bundle_id}.Internal`, resulting in a schema file named `${bundle_id}.Internal.gschema.xml`.

The app-bundle MAY assume that it will be launched with the `XDG_DATA_DIRS` environment variable[55] set to a value that includes `${prefix}/share`, so that GSettings will automatically use these schemas.

Localized strings

Some file formats, such as `.desktop` files and AppStream XML, put localized strings[56] in a single file, typically built from an international English version and a set of translations at build-time. For the following file formats, the app-bundle MUST include all of its supported translations (for example a translated `Name`) in a single file:

- Entry points
- Bundle metadata

Otherwise, application bundles that contain localized strings SHOULD use GNU gettext[57] `.mo` files. These SHOULD be stored in the `${prefix}/share/locale` hierarchy, with a subdirectory named for the *locale* in which the language is used, and a `LC_MESSAGES` subdirectory inside that containing one or more `.mo` files. The name of the `.mo` files (the *text domain*) SHOULD either be exactly the Bundle ID, or the bundle ID followed by a dot and one or more additional components. Using a single text domain whose name is exactly the bundle ID is RECOMMENDED.

[54] https://developer.gnome.org/gio/stable/GSettings.html
[55] https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html
[56] https://sjoerd.pages.apertis.org/apertis-website/architecture/internationalization/
[57] https://www.gnu.org/software/gettext/manual/index.html

For example, if the app bundle `com.example.ShoppingList` is localized into generic international French, French as spoken in Canada, and Uzbek written in Cyrillic, it might contain:

- `/Applications/com.example.ShoppingList/share/locale/fr/LC_MESSAGES/com.example.ShoppingList.mo`
- `/Applications/com.example.ShoppingList/share/locale/fr_CA/LC_MESSAGES/com.example.ShoppingList.mo`
- `/Applications/com.example.ShoppingList/share/locale/uz@cyrillic/LC_MESSAGES/com.example.ShoppingList.`

The other `LC_` directories used by gettext MAY exist alongside `LC_MESSAGES`.

If using `gettext`, programs in the app bundle would typically have to make API calls similar to these to activate these localized strings:

```
setlocale (LC_ALL, "");
bindtextdomain (GETTEXT_PACKAGE, DATADIR "/locale");
bind_textdomain_codeset (GETTEXT_PACKAGE, "UTF-8");
textdomain (GETTEXT_PACKAGE);
```

where `DATADIR` would be defined to `\"${datadir}\"` by the build system (expanded to `/Applications/com.example.ShoppingList/share` at build time), and `GETTEXT_PACKAGE` would be defined to `com.example.ShoppingList` in this example.

In general, use of gettext is not mandatory, and neither is this specific layout. Application bundles MAY store localized strings in any format of their choice, in any subdirectory of `${prefix}/lib` or `${prefix}/share`. If this is done, the application bundle author is responsible for arranging for those localized strings to be loaded.

There is one special case where use of gettext and this specific layout *is* mandatory. If an app bundle contains GSettings schemas, and those schemas support localized contents by using the `gettext-domain` attribute, then the `gettext-domain` that is declared MUST be either the Bundle ID, or the bundle ID followed by a dot and one or more additional name components. Again, using exactly the bundle ID for the gettext domain is RECOMMENDED.

> One possible direction for a future minor version would be to allow GSettings schemas to include inline translations, similar to `.desktop` files. This would require GLib modifications: at the moment this is specifically not allowed by GLib.

Generic resource data

Non-executable resource files such as graphics and sounds MUST be located in either the `${prefix}/lib` or `${prefix}/share` directory, or a descendant of one of those directories.

CPU-architecture-dependent resource files MUST be located in the

${prefix}/lib directory or a descendant of that directory. CPU-architecture-independent resource files SHOULD be located in the ${prefix}/share directory or a descendant of that directory.

The app-bundle MAY assume that it will be launched with the XDG_DATA_DIRS environment variable[58] set to a value that includes ${prefix}/share, so that any library that uses that variable (for example via g_get_system_data_dirs()) will automatically load resource files from the appropriate subdirectory of ${prefix}/share.

The app-bundle MUST NOT assume that the application framework will set environment variables that make it load resource files from ${prefix}/lib.

Example

For example, suppose the owner of example.net produces an application named Shopping List, with a graphical program to display shopping lists, and a background agent to pop up reminders when the vehicle is driven near a supermarket. Suppose the agent provides a D-Bus API to the graphical program.

Suppose this application also opens the application/vnd.example.shoppinglist content type, and handles myproduct: URIs.

Suppose the bundle ID is net.example.ShoppingList, so the bundle's files will be available at /Applications/net.example.ShoppingList. The minimal metadata required for this bundle might resemble what is shown in this section; all paths are given relative to /Applications/net.example.ShoppingList, which we will refer to as ${prefix}.

[Application bundle metadata][Bundle metadata], to be installed as ${prefix}/share/metainfo/net.example.ShoppingList.appdata.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<component type="desktop">
  <id>net.example.ShoppingList</id>
  <metadata_license>CC0-1.0</metadata_license>
  <name>Shopping List</name>
  <summary>Keep track of your groceries</summary>
  <description>
    <p>Never run out of cornflakes again with this easy-to-use shopping
      list manager, featuring:</p>
    <ul>
      <li>Special offer notifications</li>
      <li>Driving directions to the nearest supermarket</li>
      <li>Cloud synchronization</li>
    </ul>
  <developer_name>Example Software Inc.</developer_name>
```

[58] https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html

```
  <url type="homepage">https://example.net/shopping-list/</url>
  <release version="1.0" date="2016-08-23" />
</component>
```

The [settings schema][GSettings schemas] would be installed to `${prefix}/share/glib-2.0/schemas/net.example.ShoppingList.gschema.xml`, optionally accompanied by `${prefix}/share/glib-2.0/schemas/net.example.ShoppingList.enums.xml`. Those files would be compiled into `${prefix}/share/glib-2.0/schemas/gschemas.compiled`, for example by using a command like `glib-compile-schemas --strict ${DESTDIR}${prefix}/share/glib-2.0/schemas` while building the bundle.

Localized strings used in the app itself, or in its GSettings schema, would be installed as `${prefix}/share/locale/${locale}/LC_MESSAGES/com.example.ShoppingList.mo`, where `${locale}` represents a locale such as `fr_CA` or `de`.

Main entry point for the user interface, to be installed as `${prefix}/share/applications/net.example.ShoppingList`

```
[Desktop Entry]
Categories=Utility;
Exec=/Applications/net.example.ShoppingList/bin/gui
GenericName=Shopping List
Icon=net.example.ShoppingList
MimeType=application/vnd.example.shoppinglist;x-scheme-handler/myproduct;
Name=Shopping List
OnlyShowIn=Apertis;
Type=Application
X-Apertis-Type=application
X-GNOME-FullName=Example Shopping List
DBusActivatable=true
X-Apertis-ServiceExec=/Applications/net.example.ShoppingList/bin/gui      --
gapplication-service
```

The user interface's [icon][Icon for the bundle] would be installed as `${prefix}/share/icons/hicolor/64x64/apps/net.example.ShoppingList.png`.

[Entry point][Entry points] for the agent, to be installed as `${prefix}/share/applications/net.example.ShoppingLi`

```
[Desktop Entry]
Exec=/Applications/net.example.ShoppingList/bin/agent
NoDisplay=true
OnlyShowIn=Apertis;
Type=Application
X-Apertis-Type=agent-service
X-GNOME-FullName=Example Shopping List
DBusActivatable=true
```

AppArmor profile, to be installed as `${prefix}/etc/apparmor.d/Applications.net.example.ShoppingList`:

```
/Applications/net.example.ShoppingList/** {
  #include <abstractions/chaiwala-base>
  #include <abstractions/dbus-session-strict>

  /Applications/net.example.ShoppingList/{bin,libexec}/* pix,
  /Applications/net.example.ShoppingList/{bin,lib,libexec}/{,**} mr,
  /Applications/net.example.ShoppingList/share/{,**} r,

  owner /var/Applications/net.example.ShoppingList/users/** rwk,

  dbus send
    bus=session
    path=/org/freedesktop/DBus
    interface=org.freedesktop.DBus
    member={RequestName,ReleaseName}
    peer=(name=org.freedesktop.DBus),
  dbus bind bus=session name="net.example.ShoppingList",
  dbus bind bus=session name="net.example.ShoppingList.*",
  dbus (send, receive) bus=session
    peer=(label=/Applications/net.example.ShoppingList/**),
  dbus receive bus=session peer=(label=/usr/bin/canterbury),

  signal receive peer=/usr/bin/canterbury,
}
```

Future directions

Future versions of this specification could include layout and contents specifications for particular categories of system extensions[59], in particular user-installable UI themes and language packs.

Appendix: built-in application bundles

Built-in application bundles are maintained as part of the platform, and so are outside the scope of this specification. However, their structure is similar.

As a general principle, built-in application bundles that closely resemble a store application bundle, other than the structural differences listed here, will be as portable between platform versions as a similar store application bundle would be. Built-in application bundles that diverge more from that model will be more tightly-coupled to the platform for which they were designed, and so are more likely to need alterations for newer platform versions.

[59] https://sjoerd.pages.apertis.org/apertis-website/concepts/applications/#system-extensions

Structural differences

In general, built-in application bundles[60] MUST have a structure analogous to store application bundles, replacing `/Applications` with `/usr/Applications` in all path prefixes. In particular, the `${prefix}` for a built-in application bundle is `/usr/Applications/` followed by the bundle ID.

As an exception to the usual use of the `${prefix}`, built-in application bundles MUST install their [AppArmor profiles][AppArmor profile] directly to `/etc/apparmor.d`, in a file named `/etc/apparmor.d/usr.Applications.${bundle_id}` where `${bundle_id}` is to be replaced by the Bundle ID. They MUST NOT contain `/usr/Applications/*/etc/apparmor.d`.

For the following categories of files, if an equivalent store application bundle would include files in that category, built-in application bundles MUST install the real files into `${prefix}/share`. Additionally, the `.deb` file for the built-in application bundle must include symbolic links `/usr/share/*` pointing to the corresponding regular files in `${prefix}/share/*`:

- Entry points
- GSettings schemas
- Icon for the bundle
- Icons for entry points
- Bundle metadata

For example, the `.deb` file for a built-in application bundle `org.apertis.Eye` might include a symbolic link `/usr/share/applications/org.apertis.Eye.desktop` pointing to the main entry point's real file `/usr/Applications/org.apertis.Eye/share/applications/org.apertis.Eye.des` and similar symbolic links for GSettings schemas, icons and the bundle metadata.

For the following categories of files, if an equivalent store application bundle would include files in that category, built-in application bundles MUST install the files into `${prefix}`, but MUST NOT include symbolic links to them in `/usr/*`:

- Executables
- Libraries
- Icons for use by the bundle
- Localized strings
- Theme data for use by the bundle
- Generic resource data

Permissions and policy differences

Recommendations and requirements that refer to the app-store curator do not apply to built-in application bundles. The platform vendor has total control

[60] https://sjoerd.pages.apertis.org/apertis-website/concepts/applications/#software-categories

over both the platform layer[61] and the built-in application bundles that are packaged with it; they are responsible for ensuring that those components fit together correctly and meet their functional and security requirements. For example, a platform vendor can provide any AppArmor profile for a built-in application bundle, and it is up to the platform vendor to ensure that the profile is consistent with their security policy.

Graphical programs

Built-in application bundles do not necessarily need to provide their own user interfaces if they rely on an underlying service, for example one that is running in the automotive domain[62], to display a user interface. Where this specification calls for a particular entry point to be a graphical program, that requirement or recommendation does not apply to built-in application bundles. A built-in application bundle could provide similar functionality by communicating with other processes, either locally or in the automotive domain, and arranging for those other processes to display graphics instead.

However, if this is done, then the built-in application bundle is necessarily somewhat tightly coupled to the component to which it delegates its user interface.

Command line arguments

Built-in app-bundles SHOULD NOT use the `play-mode`, `app-name` or `url` tokens in their `Exec` arguments. This is a weaker prohibition than for store app-bundles, which MUST NOT use those tokens. This exception is made for backwards compatibility. Please note that the special case made for these tokens in and before Apertis 17.03 is deprecated, and their effect will change in future releases.

Agents and other non-graphical programs in built-in app-bundles SHOULD NOT have an `X-Apertis-ServiceExec` field. This is a weaker prohibition than for non-graphical programs in store app-bundles, which MUST NOT have that field: it allows those agents and non-graphical programs to make use of the special tokens like `play-mode` when run on Apertis 17.03, without including them in the service command-line. This exception is made for backwards compatibility, and is considered deprecated.

[61] https://sjoerd.pages.apertis.org/apertis-website/concepts/applications/#software-categories

[62] https://sjoerd.pages.apertis.org/apertis-website/concepts/inter-domain-communication/#automotive-domain