



Application bundle metadata

Contents

Terminology and concepts	2
Use cases	2
Audio management priorities	3
Notification and dialog priorities	3
App-bundle labelling	3
Requirements	4
App-bundle metadata	4
Labelling requirements	4
Secure identification	5
Audio stream and notification requirements	5
App-store curator oversight	5
Store app-bundle confidentiality	5
Extension points	6
Future directions	6
Other systems	7
freedesktop.org AppStream	7
Snappy	7
Android	7
Design recommendations	8
App-bundle metadata design	8
Secure identification design	9
Labelling design	9
Summary	10

This document extends the Apertis [Applications concept design](#)¹ to cover meta-data about [application bundles](#)² (app-bundles).

Terminology and concepts

See the [Apertis glossary](#)³ for background information on terminology. Apertis-specific jargon terms used in this document are hyperlinked to that glossary.

Use cases

These use-cases are not exhaustive: we anticipate that other uses will be found for per-application-bundle metadata. At the time of writing, this document concentrates on use-cases associated with assigning priorities to requests from an app-bundle to a platform service.

¹<https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application/applications/>

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

³<https://apertis-website-0b3586.pages.apertis.org/glossary/>

35 Audio management priorities

36 Assume that the Apertis [audio management](#)⁴ component assigns priorities to
37 audio streams based on [OEM](#)⁵-specific rules, potentially including user configu-
38 ration.

39 Suppose the author of an app-bundle has a legitimate reason to have their audio
40 streams played with an elevated priority, for example because their app-bundle
41 receives voice calls which should take precedence over music playback.

42 Also suppose a different, malicious app-bundle author wishes to interrupt the
43 driver's phone call to play an advertisement or other distracting sound as an
44 audio stream.

45 The Apertis system must be able to distinguish between the two app-bundles,
46 so that requests for an elevated priority from the first app-bundle can be obeyed,
47 while requests for an elevated priority from the second app-bundle are rejected.

48 We assume that the app-bundles have been checked by an app-store curator
49 before publication, and that the first app-bundle declares a special [permission](#)⁶
50 in its app manifest, resulting in the app framework allowing it to flag its audio
51 stream in ways that will result in it being treated as important, and hence
52 superseding less important audio. Conversely, if the second app-bundle had
53 declared that permission, we assume that the app-store curator would have
54 recognised this as inappropriate and reject its publication.

55 Notification and dialog priorities

56 Assume that the Apertis compositor (which is outside the scope of this docu-
57 ment) assigns priorities to notifications based on [OEM](#)⁷-specific rules, poten-
58 tially including user configuration. Depending on the OEM's chosen UX design,
59 app-modal and system-modal dialogs might be treated as visually similar to no-
60 tifications; if they are, the compositor author might also wish to assign priorities
61 from the same ranges to dialogs.

62 Similar to the [Audio management priorities](#) use case, app-bundles that have a
63 legitimate reason for their notifications or dialogs to be high-priority must be
64 able to achieve this, but malicious app-bundles whose authors aim to misuse
65 this facility must not be able to achieve an elevated priority.

66 App-bundle labelling

67 A UX designer might wish to arrange for all user interface elements associated
68 with a particular app-bundle (including notifications, windows, its representa-

⁴<https://apertis-website-0b3586.pages.apertis.org/concepts/platform/audio-management/>

⁵<https://apertis-website-0b3586.pages.apertis.org/glossary/#oem>

⁶https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_security/permissions/

⁷<https://apertis-website-0b3586.pages.apertis.org/glossary/#oem>

tion in lists of installed app-bundles, and so on) to be marked with an unambiguous indication of the app-bundle that created them, such as its name and icon.

In particular, the Compositor Security concept design (which is [work in progress](#)⁸ at the time of writing) calls for windows and notifications to be visually associated with the app-bundle that created them, so that malicious app-bundle authors cannot make the user believe that information presented by the malicious app-bundle came from a different app-bundle (*output integrity*), and also cannot convince the user to enter input into the malicious app-bundle that they had only intended to go to a different app-bundle (a *trusted input path*, providing *input confidentiality* for the non-malicious app-bundle).

Note this mechanism will not be effective unless either the app-store curator avoids accepting app-bundles with the same or confusingly similar names or icons, or the UX designer disambiguates app-bundles using something that is guaranteed to be unique, such as the app-bundle ID (which is not necessarily a desirable or user-friendly UX). This applies wherever app-bundles are listed, such as the app store's on-device user interface, the app-store's website, or a list of installed app-bundles in the device's equivalent of Android's Settings → Apps view.

Requirements

App-bundle metadata

An Apertis platform library to read app bundle metadata must be made available to platform components, featuring at least these API calls:

- given a bundle ID, return an object representing the metadata
- list all installed bundles (either built-in or store) with their IDs and metadata
- emit a signal whenever the list of installed bundles changes, for example because a store app bundle was installed, removed, upgraded or rolled back (simple change-notification)

Labelling requirements

Each app-bundle must contain a human-readable name in international English. It must also be possible for an app-bundle to contain translated versions of this name for other languages and locales, with the international English version used in locales where a translation is not provided.

Each app-bundle must be able to contain the name of the authoring company or individual.

⁸https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_security/compositor_security/

105 Each app-bundle must contain a version number. How an application developer
106 chooses to set the version numbers, however, is ultimately their decision.

107 Collabora recommends requiring version numbers to be dotted-decimal (one or
108 more decimal integers separated by single dots), with “major.minor.micro”(for
109 example 3.2.4) recommended but not strictly required.

110 **Secure identification**

111 Apertis [platform](https://apertis-website-0b3586.pages.apertis.org/glossary/#platform)⁹ services that receive requests from an unknown process must
112 be able to identify which app-bundle the process belongs to. To support this, the
113 request must take place via a channel that guarantees integrity for that process’
114 s identification: it must not be possible for a malicious process to impersonate
115 a process originating from a different app-bundle.

116 **Audio stream and notification requirements**

117 The information required by the audio manager must be represented as one or
118 more metadata key-value pairs that can be read from the app bundle metadata.

119 The information required by the notification implementation must be repre-
120 sented as one or more metadata key-value pairs that can be read from the app
121 bundle metadata.

122 We anticipate that audio management and notifications will not always assign
123 the same priority to each app-bundle, therefore it must be possible for the
124 metadata keys used by audio management and those used by notifications to be
125 distinct.

126 **App-store curator oversight**

127 It must be straightforward for an app-store curator to inspect the metadata that
128 is present in an app-bundle, for example so that they can refuse to publish app-
129 bundles that ask for audio or notification priorities that they have no legitimate
130 reason to use, or for which the name, icon or other information used for App-
131 bundle labelling is misleading.

132 **Store app-bundle confidentiality**

133 Ordinary unprivileged programs in store app-bundles must not be able to use
134 these API calls to enumerate other installed store app-bundles. For example,
135 if those API calls are implemented in terms of a D-Bus service, it must reject
136 method calls from store app-bundles, or if those API calls are implemented in
137 terms of reading the filesystem directly, store app-bundles must not be able to
138 access the necessary paths.

⁹<https://apertis-website-0b3586.pages.apertis.org/glossary/#platform>

139 *Non-requirement:* it is acceptable for ordinary unprivileged programs to be able
140 to enumerate installed built-in app-bundles. Built-in app-bundles are part of
141 the platform, so there is no expectation of confidentiality for them.

142 **Extension points**

143 We anticipate that vendors will wish to introduce non-standardized metadata,
144 either as a prototype for future standardization or to support vendor-specific
145 additional requirements. It must be possible to include new metadata fields in
146 an app-bundle, without coordination with a central authority. For example, this
147 could be achieved by namespacing new metadata fields using a DNS name ([as
148 is done in D-Bus¹⁰](#)), namespacing them with a URI ([as is done in XML¹¹](#)), or
149 using the `X-Vendor-NewMetadataField` convention¹² (as is done in email headers,
150 HTTP headers and [freedesktop.org .desktop files¹³](#)).

151 **Future directions**

152 **Platform API requirements** The application bundle metadata should in-
153 clude a minimum system version (API version) required to run the application,
154 for example to prevent the installation of an application that requires at least
155 Apertis 16.12 in an Apertis 16.09 environment. A specific versioning model for
156 the Apertis API has not yet been defined.

157 **Declaring an EULA** App-bundle metadata should include a way to specify
158 an EULA which the user must agree with before the application bundle will be
159 installed. See [AppStream issue 50¹⁴](#) for work on this topic in the AppStream
160 specification.

161 Other files in the license directory of the bundle but not mentioned in this way
162 will still be copied the device, and the HMI components must provide some way
163 to view that information later.

164 **Placeholder icons** Since the installation process is not instant, a placeholder
165 icon should be provided and specified in the version of the application bundle
166 metadata that is downloaded from the application store. This icon will be
167 copied into the store directory by the application store during publication. It
168 will be displayed by the application manager instead of the application until the
169 installation is completed. The application launcher will also be able to display
170 a progress indicator or –if multiple applications are being installed –a position
171 in the install queue.

¹⁰<https://dbus.freedesktop.org/doc/dbus-specification.html#message-protocol-names>

¹¹<https://www.w3.org/TR/REC-xml-names/>

¹²[https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.htm
l#extending](https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#extending)

¹³<https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.htm>

l

¹⁴<https://github.com/ximion/appstream/issues/50>

172 **Platform component metadata** Although it is not a requirement at this
173 stage, we anticipate that it might be useful in the future to be able to associate
174 similar metadata with platform components, such as the Newport download
175 manager.

176 Other systems

177 This section contains a very brief overview of the analogous functionality in
178 other open-source platforms.

179 freedesktop.org AppStream

180 Several open-source desktop platforms such as GNOME and KDE, and Linux
181 distributions such as Ubuntu and Fedora, have adopted [AppStream](#)¹⁵ as a
182 shared format for software component metadata, complementing the use of
183 [.desktop files](#)¹⁶ for [entry points](#)¹⁷.

184 The AppStream specification refers to *components*, which are a generalization of
185 the same concept as Apertis app-bundles, and can include software from various
186 sources, including traditional distribution packages and bundling technologies
187 such as [Flatpak](#).

188 Snappy

189 Ubuntu [Snappy](#)¹⁸ packages (snaps) are also analogous to Apertis app-bundles.
190 [Their metadata](#)¹⁹ consists of a Snappy-specific YAML file describing the snap,
191 again together with [.desktop files](#)²⁰ describing entry points.

192 Android

193 Android *apps* are its equivalent of Apertis app-bundles. Each app has a single
194 [App manifest](#)²¹ file, which is an XML file with Android-specific contents, and
195 describes both the app itself, and any *activities* that it provides (activities are
196 analogous to Apertis [entry points](#)²²).

¹⁵<https://www.freedesktop.org/software/appstream/docs/>

¹⁶<https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.htm>

1 ¹⁷https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_framework/application-entry-points/

¹⁸<http://snapcraft.io/>

¹⁹<https://snapcraft.io/docs/snapcraft-top-level-metadata>

²⁰<https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.htm>

1 ²¹<https://developer.android.com/guide/topics/manifest/manifest-intro.html>

²²https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_framework/application-entry-points/

197 Design recommendations

198 This document provides rationale for the application metadata fields, suggested
199 future directions, and details of functionality that is not necessarily long-term
200 stable.

201 App-bundle metadata design

202 We anticipate that other designs involving app-bundles will frequently require
203 other metadata beyond the use-cases currently present in this document, for
204 example categories. As such, we recommend introducing a general metadata
205 file into built-in and store app-bundles.

206 This metadata file could have any syntax and format that is readily parsed. To
207 minimize duplicate effort, we recommend using [AppStream XML](#)²³, a format
208 designed to be shared between desktop environments such as GNOME and KDE,
209 and between Linux distributions such as Ubuntu and Fedora.

210 Each app bundle (built-in or store) should install an [AppStream upstream](#)
211 [XML](#)²⁴ metadata file. If the built-in app bundle has [entry points](#)²⁵, then its
212 metadata file must be made available as `/app/share/metainfo/${bundle_id}.metainfo.xml`
213 (where `${bundle_id}` represents its bundle ID), and its `<id>` must be `<id`
214 `type="desktop">${entry_point_id}.desktop</id>` where `${entry_point_id}` repre-
215 sents its primary entry point (typically the same as the bundle ID).

216 If the app bundle has no entry points, then its metadata file must be avail-
217 able as `/app/share/metainfo/${bundle_id}.metainfo.xml` (where `${bundle_id}` rep-
218 represents its bundle ID), and its `<id>` must be the same as its bundle ID.

219 For [App-store curator oversight](#), if the implementation reads other sources
220 of metadata from a store app-bundle (for example the `.desktop` entry points
221 provided by the app-bundle), then the implementation must document those
222 sources. The app-store curator must inspect all of those sources. This require-
223 ment does not apply to built-in app-bundles, which are assumed to have been
224 checked thoroughly by the platform vendor at the time the built-in app-bundle
225 was integrated into the platform image.

226 Any metadata keys and values that have not been standardized by the App-
227 Stream project (for example audio roles that might be used to determine a
228 bundle's audio priority) must be represented using [Extension points](#) within the
229 AppStream metadata. The formal [AppStream specification](#)²⁶ does not provide
230 an extension point, but the [reference implementation](#)²⁷ and [appstream-glib](#)²⁸

²³<https://www.freedesktop.org/software/appstream/docs/>

²⁴<https://www.freedesktop.org/software/appstream/docs/chap-Metadata.html>

²⁵https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_framework/application-entry-points/

²⁶<https://www.freedesktop.org/software/appstream/docs/>

²⁷<https://www.freedesktop.org/software/appstream/docs/api/index.html>

²⁸<https://github.com/hughsie/appstream-glib/>

231 both provide support for a `<custom>` element with `<value>` children. We recom-
232 mend using that element for extension points.

233 When a store or built-in app-bundle is added, removed or changed, the Apertis
234 platform must update the corresponding cache file.

235 **Future directions** AppStream XML is equally applicable to platform com-
236 ponents, which can install metadata in `/usr/share/metainfo` in the same way as
237 built-in app-bundles.

238 Secure identification design

239 Consumers of requests from app-bundles, such as the audio manager or the
240 notifications implementation, must be able to obtain the bundle ID from the
241 request using a trusted mechanism.

242 If the request is received via D-Bus, the peer's PID must be retrieved by using
243 the [GetConnectionCredentials](#)²⁹ method call. If the request takes the form of
244 a direct `AF_UNIX` socket connection, the PID must be retrieved by reading the
245 `SO_PEERCRED` socket option. After this, the bundle ID may be obtained by parsing
246 `/proc/<PID>/root/.flatpak-info`.

247 Because the Apertis [Security concept design](#)³⁰ does not place a security bound-
248 ary between different processes originating from the same app-bundle, all identi-
249 fication of app-bundles should be carried out using their bundle IDs. In particu-
250 lar, consumers of requests from app-bundles should only use the requester's PID
251 to derive its bundle ID and whether it is a store or built-in app-bundle, and must
252 not use the complete path of the executable or the name of the corresponding
253 [entry point](#)³¹ in access-control decisions.

254 Labelling design

255 [AppStream upstream XML](#)³² already contains standardized metadata fields for
256 a name, author name etc.

257 The name (and several other metadata fields) can be translated via the `xml:lang`
258 attribute. For example, GNOME Videos (Totem) has many language-specific
259 names, starting with:

```
260 <name>Videos</name>  
261 <name xml:lang="af">Video's</name>  
262 <name xml:lang="ar">فيديو</name>
```

²⁹<https://dbus.freedesktop.org/doc/dbus-specification.html#bus-messages-get-connection-credentials>

³⁰https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_security/security/

³¹https://apertis-website-0b3586.pages.apertis.org/concepts/archive/application_framework/application-entry-points/

³²<https://www.freedesktop.org/software/appstream/docs/chap-Metadata.html>

```

263 <name xml:lang="as">𑄧𑄭𑄭𑄭𑄭'𑄧𑄭𑄭𑄭</name>
264 <name xml:lang="be">Бiдэа</name>

```

AppStream upstream XML does not include an icon, although the derived [AppStream catalog XML](#)³³ format published by redistributors does. We recommend that the app-bundle should contain a PNG icon whose name matches its bundle ID, installed to its `share/` directory as part of the `hicolor` fallback theme.

The reserved icon theme name `hicolor` is used as the fallback whenever a specific theme does not have the required icon, as specified in the [freedesktop.org Icon Theme specification](#)³⁴. The name `hicolor` was chosen for historical reasons.

For example, `com.example.ShoppingList` would include `/app/share/icons/hicolor/64x64/apps/com.example.ShoppingList.png`. If the app-store uses AppStream catalog XML, then the process used to build AppStream catalog XML from individual applications' AppStream upstream XML files should assume this icon name and include it in the catalog XML.

Open question: We should require a specific size for the icon, to avoid blurry or blocky app icons caused by resizing. GNOME Software uses 64×64 as its baseline requirement, but recommends larger icons, for example 256×256 . [iOS](#)³⁵ uses 1024×1024 for the App Store and ranges from 60×60 to 180×180 for on-device icons. [Android][Android icons sizes] uses 512×512 for the Google Play Store and ranges from 36×36 to 96×96 for on-device icons. What are our preferred sizes?

Future directions Platform components that are not part of an app-bundle do not have bundle IDs. We anticipate that **Platform component metadata** might be identified by a separate identifier in the same reversed-DNS namespace, and that the consumer of requests might derive the platform component identifier by looking for components that declare metadata fields matching the requester's AppArmor label (part of the AppArmor context).

Summary

- **Secure identification** is provided by using the peer's PID to determine the bundle ID.
- The **Audio stream and notification requirements** are addressed by providing their desired metadata in the app-bundle metadata, in the form of arbitrary key/value pairs.
- **App-store curator oversight** is facilitated by documenting all of the sources within a store app-bundle from which the implementation gathers metadata to populate its cache.

³³<https://www.freedesktop.org/software/appstream/docs/chap-CatalogData.html>

³⁴<http://standards.freedesktop.org/icon-theme-spec/icon-theme-spec-latest.html>

³⁵<https://developer.apple.com/library/safari/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html>

- 299 • **Store app-bundle confidentiality** is provided by storing the cache file de-
300 scribing installed store app-bundles in a location where store app-bundles
301 cannot read it, and by avoiding the need to introduce a D-Bus service
302 from which they could obtain the same information.
- 303 • The `appstream-glib`³⁶ library supports **Extension points** in AppStream
304 XML.

³⁶<https://github.com/hughsie/appstream-glib/>