



Application bundle metadata

1 **Contents**

2 Terminology and concepts 2
3 Use cases 2
4 Audio management priorities 2
5 Notification and dialog priorities 3
6 App-bundle labelling 3
7 Requirements 3
8 App-bundle metadata 3
9 Labelling requirements 4
10 Secure identification 4
11 Audio stream and notification requirements 5
12 App-store curator oversight 5
13 Store app-bundle confidentiality 5
14 Extension points 5
15 Future directions 6
16 Other systems 7
17 freedesktop.org AppStream 7
18 Snappy 8
19 Android 8
20 Design recommendations 8
21 App-bundle metadata design 8
22 Secure identification design 11
23 Labelling design 12
24 Summary 13

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

27 **Terminology and concepts**

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

30 **Use cases**

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

¹<https://em.pages.apertis.org/apertis-website/concepts/applications/>

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

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

35 **Audio management priorities**

36 Assume that the Apertis audio management component (which is outside the
37 scope of this document) assigns priorities to audio streams based on OEM⁴-
38 specific rules, potentially including user configuration.

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, so
46 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](#)⁵
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 doc-
57 ument) assigns priorities to notifications based on OEM⁶-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-
69 tion in lists of installed app-bundles, and so on) to be marked with an unam-

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

⁵<https://em.pages.apertis.org/apertis-website/concepts/permissions/>

⁶<https://em.pages.apertis.org/apertis-website/glossary/#oem>

70 biguous indication of the app-bundle that created them, such as its name and
71 icon.

72 In particular, the Compositor Security concept design (which is [work in](#)
73 [progress](#)⁷ at the time of writing) calls for windows and notifications to be
74 visually associated with the app-bundle that created them, so that malicious
75 app-bundle authors cannot make the user believe that information presented by
76 the malicious app-bundle came from a different app-bundle (*output integrity*),
77 and also cannot convince the user to enter input into the malicious app-bundle
78 that they had only intended to go to a different app-bundle (a *trusted input*
79 *path*, providing *input confidentiality* for the non-malicious app-bundle).

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

88 **Requirements**

89 **App-bundle metadata**

90 An Apertis platform library to read app bundle metadata must be made avail-
91 able to platform components, featuring at least these API calls:

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

98 **Labelling requirements**

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

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

105 Each app-bundle must contain a version number. To let the application manager
106 make appropriate decisions, all application bundles must a format for their

⁷https://em.pages.apertis.org/apertis-website/concepts/compositor_security/

107 version strings that can be compared in a standard way. How an application
108 developer chooses to set the version numbers is ultimately their decision, but
109 Apertis must be able to determine whether one version number is higher than
110 another.

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

114 There will be a “store version” appended to the version string after a dash,
115 similar to the versioning scheme used by `dpkg`; for example, in the version string
116 3.2.4-1, the 1 is the store version. The store version allows the store to push an
117 update even if the application version hasn’t changed, and it will be the lowest
118 significant figure. For example, version 2.2.0-1 is newer than version 2.1.99-4.
119 The store version will re-start at 1 any time the application version is increased,
120 and will be incremented if a new intermediate release is required.

121 **Secure identification**

122 Apertis [platform](#)⁸ services that receive requests from an unknown process must
123 be able to identify which app-bundle the process belongs to. To support this, the
124 request must take place via a channel that guarantees integrity for that process’s
125 identification: it must not be possible for a malicious process to impersonate a
126 process originating from a different app-bundle.

127 **Audio stream and notification requirements**

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

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

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

137 **App-store curator oversight**

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

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

143 **Store app-bundle confidentiality**

144 Ordinary unprivileged programs in store app-bundles must not be able to use
145 these API calls to enumerate other installed store app-bundles. For example,
146 if those API calls are implemented in terms of a D-Bus service, it must reject
147 method calls from store app-bundles, or if those API calls are implemented in
148 terms of reading the filesystem directly, store app-bundles' AppArmor profiles
149 must not allow reading the necessary paths.

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

153 **Extension points**

154 We anticipate that vendors will wish to introduce non-standardized metadata,
155 either as a prototype for future standardization or to support vendor-specific
156 additional requirements. It must be possible to include new metadata fields in
157 an app-bundle, without coordination with a central authority. For example, this
158 could be achieved by namespacing new metadata fields using a DNS name ([as
159 is done in D-Bus⁹](#)), namespacing them with a URI ([as is done in XML¹⁰](#)), or
160 using the `X-Vendor-NewMetadataField` convention¹¹ (as is done in email headers,
161 HTTP headers and [freedesktop.org .desktop files¹²](#)).

162 **Future directions**

163 **Platform API requirements**

164 The application bundle metadata should include a minimum system version
165 (API version) required to run the application, for example to prevent the instal-
166 lation of an application that requires at least Apertis 16.12 in an Apertis 16.09
167 environment. A specific versioning model for the Apertis API has not yet been
168 defined.

169 **Declarative permissions**

170 The application bundle metadata should include simple, declarative [permis-
171 sions¹³](#) which can be used to generate an AppArmor profile in an automated
172 way. The [Permissions concept design¹⁴](#) tracks this work.

⁹<https://dbus.freedesktop.org/doc/dbus-specification.html#message-protocol-names>

¹⁰<https://www.w3.org/TR/REC-xml-names/>

¹¹[https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.
html#extending](https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#extending)

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

¹³<https://em.pages.apertis.org/apertis-website/concepts/permissions/>

¹⁴<https://em.pages.apertis.org/apertis-website/concepts/permissions/>

173 **Declaring system extensions**

174 The [Applications concept design](#)¹⁵ calls for app-bundle metadata to describe
175 the types of [system extension](#)¹⁶ (themes, addons, plugins, etc.) provided by an
176 app-bundle. There is currently no detailed specification for this.

177 AppStream upstream XML already supports declaring that a component (app-
178 bundle) is an addon to another component (via the `addon` type) or to the system
179 as a whole (via the `<provides>` element). There is no specific metadata to de-
180 scribe themes; discussion has been started in [AppStream issue 67](#)¹⁷.

181 **Declaring where applications store non-essential files**

182 The [Applications concept design](#)¹⁸ suggests that application bundle metadata
183 might declare where applications store non-essential files, so that the system
184 can delete those files when disk space runs out.

185 **Placing symbolic links in centralized locations**

186 The [Applications concept design](#)¹⁹ suggests that for applications not
187 originally designed for Apertis, which might write to locations like
188 `~/.someapp`, application bundle metadata might declare where the plat-
189 form must create symbolic links to cause those applications to read and
190 write more appropriate locations on Apertis, for example `~/.someapp →`
191 `/var/Applications/com.example.SomeApp/users/${UID}/data`.

192 **Declaring an EULA**

193 App-bundle metadata should include a way to specify an EULA which the user
194 must agree with before the application bundle will be installed. See [AppStream](#)
195 [issue 50](#)²⁰ for work on this topic in the AppStream specification.

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

199 **Placeholder icons**

200 Since the installation process is not instant, a placeholder icon should be pro-
201 vided and specified in the version of the application bundle metadata that is
202 downloaded from the application store. This icon will be copied into the store
203 directory by the application store during publication. It will be displayed by

¹⁵<https://em.pages.apertis.org/apertis-website/concepts/applications/>

¹⁶<https://em.pages.apertis.org/apertis-website/concepts/applications/#system-extensions>

¹⁷<https://github.com/ximion/appstream/issues/67>

¹⁸<https://em.pages.apertis.org/apertis-website/concepts/applications/>

¹⁹<https://em.pages.apertis.org/apertis-website/concepts/applications/>

²⁰<https://github.com/ximion/appstream/issues/50>

204 the application manager instead of the application until the installation is com-
205 pleted. The application launcher will also be able to display a progress indicator
206 or – if multiple applications are being installed – a position in the install queue.

207 **Platform component metadata**

208 Although it is not a requirement at this stage, we anticipate that it might be
209 useful in the future to be able to associate similar metadata with platform
210 components, such as the Newport download manager.

211 **Other systems**

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

214 **freedesktop.org AppStream**

215 Several open-source desktop platforms such as GNOME and KDE, and Linux
216 distributions such as Ubuntu and Fedora, have adopted [AppStream](#)²¹ as a
217 shared format for software component metadata, complementing the use of
218 [.desktop files](#)²² for [entry points](#)²³.

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

223 **Flatpak**

224 The [Flatpak](#)²⁴ framework provides user-installable applications analogous to
225 Apertis app-bundles. It uses [AppStream](#)²⁵ for app-bundle metadata, together
226 with [.desktop files](#)²⁶ for entry points.

227 **Snappy**

228 Ubuntu [Snappy](#)²⁷ packages (snaps) are also analogous to Apertis app-bundles.
229 [Their metadata](#)²⁸ consists of a Snappy-specific YAML file describing the snap,

²¹<https://www.freedesktop.org/software/appstream/docs/>

²²<https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html>

²³<https://em.pages.apertis.org/apertis-website/concepts/application-entry-points/>

²⁴<http://flatpak.org/>

²⁵<https://www.freedesktop.org/software/appstream/docs/>

²⁶<https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html>

²⁷<http://snapcraft.io/>

²⁸<https://snapcraft.io/docs/snapcraft-top-level-metadata>

230 again together with `.desktop files`²⁹ describing entry points.

231 **Android**

232 Android *apps* are its equivalent of Apertis app-bundles. Each app has a single
233 `App manifest`³⁰ file, which is an XML file with Android-specific contents, and
234 describes both the app itself, and any *activities* that it provides (activities are
235 analogous to Apertis `entry points`³¹).

236 **Design recommendations**

237 The `Apertis Application Bundle Specification`³² describes the metadata fields
238 that can appear in an application bundle and are expected to remain supported
239 long-term. This document provides rationale for those fields, suggested future
240 directions, and details of functionality that is not necessarily long-term stable.

241 **App-bundle metadata design**

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

246 This metadata file could have any syntax and format that is readily parsed. To
247 minimize duplicate effort, we recommend using `AppStream XML`³³, a format
248 designed to be shared between desktop environments such as GNOME and KDE,
249 and between Linux distributions such as Ubuntu and Fedora.

250 Each built-in app bundle should install an `AppStream upstream XML`³⁴
251 metadata file. If the built-in app bundle has `entry points`³⁵, then its metadata
252 file must be made available as `/usr/share/metainfo/${bundle_id}.appdata.xml`
253 (where `${bundle_id}` represents its bundle ID), and its `<id>` must be `<id`
254 `type="desktop">${entry_point_id}.desktop</id>` where `${entry_point_id}` rep-
255 represents its primary entry point (typically the same as the bundle ID).
256 `/usr/share/metainfo/${bundle_id}.appdata.xml` will typically be a symbolic link
257 to `/usr/Applications/${bundle_id}/share/metainfo/${bundle_id}.appdata.xml`.

258 If the built-in app bundle has no entry points (for example a theme), then its
259 metadata file must be available as `/usr/share/metainfo/${bundle_id}.metainfo.xml`
260 (where `${bundle_id}` represents its bundle ID), and its `<id>` must be the same as

²⁹<https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html>

³⁰<https://developer.android.com/guide/topics/manifest/manifest-intro.html>

³¹<https://em.pages.apertis.org/apertis-website/concepts/application-entry-points/>

³²<https://em.pages.apertis.org/apertis-website/architecture/bundle-spec/>

³³<https://www.freedesktop.org/software/appstream/docs/>

³⁴<https://www.freedesktop.org/software/appstream/docs/chap-Metadadata.html>

³⁵<https://em.pages.apertis.org/apertis-website/concepts/application-entry-points/>

261 its bundle ID. Again, this would typically be a symbolic link to a corresponding
262 path in `/usr/Applications/${bundle_id}`.

263 Each store app bundle should install an AppStream upstream XML metadata
264 file into `/Applications/${bundle_id}/share/metainfo/${bundle_id}.appdata.xml`
265 or `/Applications/${bundle_id}/share/metainfo/${bundle_id}.metainfo.xml` (de-
266 pending on whether it has entry points), with contents corresponding to those
267 specified for built-in app bundles. For **Store app-bundle confidentiality**, a store
268 app-bundle's AppArmor profile must not allow it to read the contents of a
269 different store app-bundle, and in particular its AppStream metadata.

270 AppStream upstream XML is normally also searched for in the **depre-**
271 **cated path**³⁶ `/usr/share/appdata`, but for simplicity, we do not require the
272 `share/appdata/` directory to be processed for application bundles. Since existing
273 application bundles do not contain it, this does not create a compatibility
274 problem.

275 For **App-store curator oversight**, if the implementation reads other sources
276 of metadata from a store app-bundle (for example the `.desktop` entry points
277 provided by the app-bundle), then the implementation must document those
278 sources. The app-store curator must inspect all of those sources. This require-
279 ment does not apply to built-in app-bundles, which are assumed to have been
280 checked thoroughly by the platform vendor at the time the built-in app-bundle
281 was integrated into the platform image.

282 The Apertis platform must provide cache files whose timestamps change when-
283 ever there is a change to the set of store or built-in app bundles, or to those
284 bundles' contents. These cache files should be monitored by the **libcanterbury-**
285 **platform**³⁷ library, using the standard `inotify` mechanism. Any cache files that
286 contain store app-bundles must not be readable by a store app-bundle, to pre-
287 serve **Store app-bundle confidentiality**.

288 The other APIs that are required are straightforward to implement in the
289 **libcanterbury-platform**³⁸ library by reading from the cache files. Because this is
290 done in a library (as opposed to a D-Bus service), the implementation of these
291 APIs will run with the privileges of the process that is requesting the informa-
292 tion: in particular, if an unprivileged process attempts to read the cache files,
293 this will be prevented by its AppArmor profile, regardless of whether it is using
294 **libcanterbury-platform** or reading the files directly.

295 We recommend that store app-bundles and built-in app-bundles appear in sep-
296 arate cache files, for several reasons:

- 297 • In the current design for Apertis operating system upgrades, the
298 metadata files for built-in app-bundles and platform components in

³⁶<https://www.freedesktop.org/software/appstream/docs/chap-Metadata.html#spec-component-location>

³⁷<https://gitlab.apertis.org/appfw/canterbury>

³⁸<https://gitlab.apertis.org/appfw/canterbury>

299 /usr/Applications/*/share/* and /usr/share/* are only updated during
300 an operating system upgrade, by either `dpkg` or by unpacking a new
301 OS filesystem hierarchy that will be activated after the next reboot.
302 In the `dpkg` case, it is sufficient to have a `dpkg` trigger monitoring these
303 directories, and update the built-in app-bundle cache when they have
304 changed, leaving the store app-bundle cache unchanged. Similarly, in the
305 whole-OS upgrade case, the built-in app-bundle cache can be provided in
306 the new OS filesystem or rebuilt during startup, again leaving the store
307 app-bundle cache unchanged.

- 308 • Conversely, the metadata files for store app-bundles are updated by the
309 Ribchester subvolume manager when it installs a new store app-bundle,
310 which can occur at any time. When it does this, it is sufficient to up-
311 date the store app-bundle cache, leaving the built-in app-bundle cache
312 unchanged.
- 313 • If Apertis moves to a more static model for deployment of the platform
314 (for example using disk images or OSTree to deploy pre-built filesystem
315 hierarchies), the built-in app-bundle cache would be entirely static and
316 could be included in the pre-built filesystem hierarchy.
- 317 • Using separate caches makes it straightforward to ensure that if a store
318 app-bundle with the same name as a built-in app-bundle is somehow in-
319 stalled, the built-in app-bundle takes precedence.

320 Any metadata keys and values that have not been standardized by the App-
321 Stream project (for example audio roles that might be used to determine a
322 bundle's audio priority) must be represented using **Extension points** within the
323 AppStream metadata. The formal **AppStream specification**³⁹ does not provide
324 an extension point, but the **reference implementation**⁴⁰ and **appstream-glib**⁴¹
325 both provide support for a `<custom>` element with `<value>` children. We re-
326 commend using that element for extension points. See the **Apertis Application**
327 **Bundle Specification**⁴² for details.

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

330 **Future directions**

331 AppStream XML is equally applicable to platform components, which can install
332 metadata in `/usr/share/metainfo` in the same way as built-in app-bundles.

333 Because built-in app-bundles and platform components have the same update
334 schedule and are managed by the same vendor (they are both part of the plat-

³⁹<https://www.freedesktop.org/software/appstream/docs/>

⁴⁰<https://www.freedesktop.org/software/appstream/docs/api/index.html>

⁴¹<https://github.com/hughsie/appstream-glib/>

⁴²<https://em.pages.apertis.org/apertis-website/architecture/bundle-spec/>

335 form), we anticipate that platform components should use the same cache file
336 as built-in app-bundles.

337 **Secure identification design**

338 Consumers of requests from app-bundles, such as the audio manager or the
339 notifications implementation, must receive the bundle ID alongside the request
340 using a trusted mechanism. If the request is received via D-Bus, the bundle
341 ID must be retrieved by using the `GetConnectionCredentials`⁴³ method call to
342 receive the AppArmor context, then parsing the context to get the bundle ID
343 and whether it is a store or built-in app-bundle. If the request takes the form of
344 a direct `AF_UNIX` socket connection, the bundle ID must be retrieved by reading
345 the `SO_PEERCRED` socket option, then parsed in the same way. Consumers of app-
346 bundle priorities should do this by using the `CbyProcessInfo`⁴⁴ objects provided
347 by `libcanterbury`⁴⁵.

348 Because the Apertis [Security concept design](#)⁴⁶ does not place a security bound-
349 ary between different processes originating from the same app-bundle, all iden-
350 tification of app-bundles should be carried out using their bundle IDs. In par-
351 ticular, consumers of requests from app-bundles should only use the requester's
352 AppArmor label to derive its bundle ID and whether it is a store or built-in app-
353 bundle, and must not use the complete AppArmor label, the complete path of
354 the executable or the name of the corresponding [entry point](#)⁴⁷ in access-control
355 decisions.

356 **Labelling design**

357 [AppStream upstream XML](#)⁴⁸ already contains standardized metadata fields for
358 a name, author name etc.

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

```
362 <name>Videos</name>  
363 <name xml:lang="af">Video's</name>  
364 <name xml:lang="ar">فيديو</name>  
365 <name xml:lang="as">বীডিও</name>  
366 <name xml:lang="be">Відэа</name>
```

367 AppStream upstream XML does not include an icon, although the derived [App-](#)

⁴³<https://dbus.freedesktop.org/doc/dbus-specification.html#bus-messages-get-connection-credentials>

⁴⁴<https://gitlab.apertis.org/appfw/canterbury/blob/master/canterbury/process-info.h>

⁴⁵<https://gitlab.apertis.org/appfw/canterbury>

⁴⁶<https://em.pages.apertis.org/apertis-website/concepts/security/>

⁴⁷<https://em.pages.apertis.org/apertis-website/concepts/application-entry-points/>

⁴⁸<https://www.freedesktop.org/software/appstream/docs/chap-Metadata.html>

368 [Stream collection XML](#)⁴⁹ format published by redistributors does. We recom-
369 mend that the app-bundle should contain a PNG icon whose name matches its
370 bundle ID, installed to its `share/` directory as part of the `hicolor` fallback theme.

371 The reserved icon theme name `hicolor` is used as the fallback when-
372 ever a specific theme does not have the required icon, as specified in
373 the [freedesktop.org Icon Theme specification](#)⁵⁰. The name `hicolor`
374 was chosen for historical reasons.

375 For example, `com.example.ShoppingList` would include `/Applications/com.example.ShoppingList/share/icons/hicolor`
376 If the app-store uses AppStream collection XML, then the process used to build
377 AppStream collection XML from individual applications' AppStream upstream
378 XML files should assume this icon name and include it in the collection XML.

379 **Open question:** We should require a specific size for the icon, to avoid blurry
380 or blocky app icons caused by resizing. GNOME Software uses 64×64 as its
381 baseline requirement, but recommends larger icons, for example 256×256. [iOS](#)⁵¹
382 uses 1024×1024 for the App Store and ranges from 60×60 to 180x180 for on-
383 device icons. [Android][Android icons sizes] uses 512×512 for the Google Play
384 Store and ranges from 36×36 to 96×96 for on-device icons. What are our
385 preferred sizes?

386 Future directions

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

393 Summary

- 394 • [app-bundle metadata](#) is read from the cache that summarizes built-in and
395 store app-bundles. The [libcanterbury-platform](#)⁵² library provides the re-
396 quired APIs; in particular, change notification can be done using `inotify`.
- 397 • [Secure identification](#) is provided by [parsing the requesting process's Ap-
398 pArmor context][Secure identification design].
- 399 • The [Audio stream and notification requirements](#) are addressed by provid-
400 ing their desired metadata in the app-bundle metadata, in the form of
401 arbitrary key/value pairs.

⁴⁹<https://www.freedesktop.org/software/appstream/docs/chap-CollectionData.html>

⁵⁰<http://standards.freedesktop.org/icon-theme-spec/icon-theme-spec-latest.html>

⁵¹<https://developer.apple.com/library/safari/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html>

⁵²<https://gitlab.apertis.org/appfw/canterbury>

- 402 • **App-store curator oversight** is facilitated by documenting all of the sources
403 within a store app-bundle from which the implementation gathers meta-
404 data to populate its cache.
- 405 • **Store app-bundle confidentiality** is provided by storing the cache file de-
406 scribing installed store app-bundles in a location where store app-bundles
407 cannot read it, and by avoiding the need to introduce a D-Bus service
408 from which they could obtain the same information.
- 409 • The [appstream-glib](#)⁵³ library supports **Extension points** in AppStream
410 XML.

⁵³<https://github.com/hughsie/appstream-glib/>