Adding Applications to the GNOME Software Center

Abstract

Traditionally we have had little information about Linux applications before they have been installed. With the creation of a software center we require access to rich set of metadata about an application **before it is deployed** so it it can be displayed to the user and easily installed.

The AppStream specification is an mature and evolving standard that allows upstream applications to provide metadata such as localized descriptions, screenshots, extra keywords and content ratings for parental control. The basic concept is that the upstream project ships one extra AppData XML file which is used to build a global application catalog called an AppStream file. Over 1000 upstream projects now include AppData files, and the software center shipped in Fedora, Ubuntu and OpenSuse is now an easy to use application filled with useful application metadata. Applications without AppData files are no longer shown which provides quite some incentive to upstream projects wanting visibility in popular desktop environments.

Revisions

Revision 0.9 14th October 2016

Introduction

Installing applications on Linux has traditionally involved copying binary and data files into a directory and just writing a single desktop file into a per-user or per-system directory so that it shows up in the desktop environment. In this document we refer to applications as graphical programs, rather than other system add-on components like drivers and codecs. This document will explain why the extra metadata is required and what is required for an application to be visible in the software center, both as a rpm package and as a flatpak bundle.

System Architecture

Linux File Hierarchy

Applications on Linux are expected to install binary files to `/usr/bin`, the install architecture independent data files to `/usr/share/` and configuration files to `/etc`. Small temporary files can be stored in `/tmp` and much larger files in /var/tmp. Per-user configuration is either stored in the users home directory (in `~/.config`) or stored in a binary settings store such as dconf.

See the File Hierarchy Standard[4] for more information.

Desktop files

The creation of a desktop file on Linux allows a program to be visible to the graphical environment, e.g. KDE or GNOME Shell. If applications do not have a desktop file they must be manually launched using a terminal emulator. Desktop files must adhere to the Desktop File Specification[1] and provide metadata in an ini-style format such as:

- Binary type, typically ‘`Application`’
- Program name (optionally localized)
- Icon to use in the desktop shell
- Program binary name to use for launching
- Any mime types that can be opened by the applications (optional)
- The standard categories the application should be included in (optional)
- Keywords (optional, and optionally localized)
- Short one-line summary (optional, and optionally localized)

The desktop file would be found in `/usr/share/applications` if installed for all users:

```
[Desktop Entry]
Type=Application
Name=OpenSCAD
Icon=openscad
Exec=openscad %f
MimeType=application/x-openscad;
Categories=Graphics;3DGraphics;Engineering;
Keywords=3d;solid;geometry;csg;model;stl;
```

Text 1: example desktop file for the OpenScad project

The desktop files are used when creating the software center metadata, and so you should verify that you ship a .desktop file for each built application, and that these keys exist: Name, Comment, Icon, Categories, Keywords and Exec and that desktop-file-validate correctly validates the file. There should also be only one desktop file for each application.

The application icon should be in the PNG format with a transparent background and installed in `/usr/share/icons`, `/usr/share/icons/hicolor/*/apps/*`, or `/usr/share/${app_name}/icons/*`. The icon should be at least 64×64 in size.

The file name of the desktop file is also very important, as this is the assigned ‘application ID’. New applications typically use a reverse-DNS style, e.g. `org.gnome.Nautilus.desktop` but older programs may just use a short name, e.g. `gimp.desktop`. It is important to note that the file extension **is also included** as part of the desktop ID.

AppData Files

At least one valid AppData file with the suffix `.appdata.xml` file should be installed into `/usr/share/appdata` with an `<id>` that matches the name of the .desktop file, e.g. `gimp.appdata.xml` or `org.gnome.Nautilus.appdata.xml`.

In the AppData file you should include several 16:9 aspect screenshots along with a compelling translated description made up of multiple paragraphs. Make sure you follow the style guide, which can be tested using `appstream-util validate foo.appdata.xml`

What is allowed in an AppData file is defined in the AppStream specification[2] but common items typical applications add is:

- License of the upstream project in SPDX, or ‘Proprietary’
- A translated name and short description to show in the software center search results
- A translated long description, consisting of multiple paragraphs, itemized and ordered lists.

- A number of screenshots, with localized captions, typically in 16:9 aspect ratio – these will typically mirrored before
- An optional list of releases with the update details and release information.
- An optional list of kudos which tells the software center about the integration level of the application
- A set of URLs that allow the software center to provide links to help or bug information
- An optional gettext or QT translation domain which allows the AppStream generator to collect statistics on shipped application translations.

A typical (albeit somewhat truncated) AppData file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<component type="desktop">
  <id>org.gnome.MultiWriter.desktop</id>
  <metadata_license>CC0-1.0</metadata_license>
  <project_license>GPL-2.0+</project_license>
  <name>MultiWriter</name>
  <name xml:lang="bs">Grupni pisač</name>
  <summary>Write an ISO file to multiple USB devices at once</summary>
  <summary xml:lang="bs">Zapišite ISO datoteku na nekoliko USB uređaja odjednom</summary>
  <description>
     <p>GNOME MultiWriter can be used to write an ISO file to multiple USB devices at once. Supported
drive sizes are between 1GB and 32GB.</p>
    <p xml:lang="bs">Gnomov Grupni pisač može da se koristi za zapisivanje ISO datoteke na nekoliko USB
uređaja odjednom. Podržane veličine diskova su od 1GB do 32GB.</p>
  </description>
  <screenshots>
    <screenshot type="default">
      <image>https://git.gnome.org/browse/gnome-multi-writer/plain/data/appdata/gmw-startup.png</image>
      <caption>Initial screen for the application</caption>
      <caption xml:lang="bs">Početni ekran programa</caption>
    </screenshot>
  </screenshots>
  <releases>
    <release version="3.20.0" date="2016-03-21">
      <description>
        <p>This is the first stable release for GNOME 3.20</p>
      </description>
    </release>
  </releases>
  <kudos>
    <kudo>AppMenu</kudo>
    <kudo>HiDpiIcon</kudo>
    <kudo>ModernToolkit</kudo>
  </kudos>
  <url type="homepage">https://wiki.gnome.org/Apps/MultiWriter</url>
  <url type="bugtracker">https://bugzilla.gnome.org/browse.cgi?product=gnome-multi-writer</url>
  <translation type="gettext">gnome-multi-writer</translation>
</component>
```

AppStream Metadata

AppStream[2] was first discussed in 2008 and since then many people have contributed to the specification. It is being used primarily for application metadata but also now is used for drivers, firmware, input methods and fonts. There are multiple projects producing AppStream metadata and also a number of projects consuming the final XML metadata.

When applications are being built as packages by a distribution then the AppStream generation is done automatically, and you do not need to do anything other than installing a .desktop file and an appdata.xml file in the upstream tarball or zip file.

If the application is being built externally then the distributor will need to generate the AppStream metadata manually. This would be used when internal-only or closed source software is being either used or produced. This document assumes you are currently building RPM packages and exporting yum-style repository metadata for Fedora or RHEL although the concepts are the same for rpm-on-OpenSuse or deb-on-Ubuntu.

NOTE: If you are building packages, make sure that there are not two applications installed with one single package. If this is currently the case split up the package so that there are multiple subpackages or mark one of the .desktop files as NoDisplay=true. Make sure the application-subpackages depend on any -common subpackage and deal with upgrades (perhaps using a metapackage) if you've shipped the application before.

Yum Metadata:

When GNOME Software checks for updates it downloads various metadata files from the server describing the packages available in the repository. GNOME Software can also download AppStream metadata at the same time, allowing add-on repositories to include applications that are visible in the the software center.

In most cases distributors are already building binary RPMS and then building metadata as an additional step by running something like this to generate the repomd files on a directory of packages:

```
$ createrepo_c --no-database --simple-md-filenames SRPMS/
$ createrepo_c --no-database --simple-md-filenames x86_64/
```

This creates the primary and filelist metadata required for updating on the command line. To build the metadata required for the software center we we need to actually generate the AppStream XML. This works by decompressing `.rpm` files and merging together the `.desktop` file, the `.appdata.xml` file and preprocessing the icons. Remember, only applications installing AppData files will be included in the metadata.

```
$ appstream-builder                \
     --origin=yourcompanyname      \
     --basename=appstream          \
     --cache-dir=/tmp/asb-cache    \
     --enable-hidpi                \
     --max-threads=1               \
     --min-icon-size=32            \
     --output-dir=/tmp/asb-md      \
     --packages-dir=x86_64/        \
     --temp-dir=/tmp/asb-icons
```

This takes a few minutes and generates some files to the output directory. The actual build output will depend on your compose server configuration. At this point you can also verify the application is visible in the `yourcompanyname.xml.gz` file.

We then have to take the generated XML and the tarball of icons and add it to the repomd.xml master document so that GNOME Software automatically downloads the content for searching. This is as simple as doing:

```
modifyrepo_c                       \
     --no-compress                 \
     --simple-md-filenames         \
     /tmp/asb-md/appstream.xml.gz  \
     x86_64/repodata/
modifyrepo_c                       \
     --no-compress                 \
     --simple-md-filenames         \
     /tmp/asb-md/appstream-icons.tar.gz \
     x86_64/repodata/
```

Deploying this metadata like the other files will allow GNOME Software to add the application metadata the next time the repository is refreshed, typically, once per day.

Flapak Metadata

The flatpak-builder binary generates AppStream metadata automatically when building applications if the `appstream-compose` tool is installed on the flatpak build machine. Flatpak remotes are exported with a separate ‘appstream’ branch which is automatically downloaded by GNOME Software and no addition work if required when building your application or updating the remote. Adding the remote is enough to add the application to the software center, on the assumption the AppData file is valid.

Conclusions

AppStream files allow us to build a modern software center experience either using legacy distro packages with yum-style metadata or with the new flatpak application deployment framework. By including a desktop file and AppData file for your Linux binary build your application can be easily installed by end users.

Future Work

AppData currently uses the OARS content rating system which will be expanded for more uses cases and filtering options.

Related Work

The ODRS[3] is a web service which provides end-user moderated application reviews using the AppStream application ID.

Acknowledgments

Much gratitude has to go to Red Hat for funding my work on this for the last few years. I have to also thank all the early-adopter projects that took a leap of faith for the software center I was trying to achieve.

Citations

1. Desktop Specification: 17th October 2016:

   https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#introduction

2. AppStream Specification 0.10: 17th October 2016:

   https://www.freedesktop.org/software/appstream/docs/

3. Open Desktop Review System: 17th October 2016:

   https://odrs.gnome.org/

4. Filesystem Hierarchy Standard: 17th October 2016:

   https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard