# notify2 Documentation

*Release 0.3*

**Thomas Kluyver**

**Apr 28, 2017**

# Contents

notify2 is a replacement for pynotify which can be used from different GUI toolkits and from programs without a GUI. The API is largely the same as that of pynotify, but some less important parts are left out.

You can alternatively use the GObject Introspection bindings to libnotify (`from gi.repository import Notify`). I'd recommend that for GTK applications, while notify2 has fewer dependencies for non-GTK applications. It should be easy to switch between the two.

Notifications are sent to a notification daemon over D-Bus, according to the Desktop notifications spec, and the server is responsible for displaying them to the user. So your application has limited control over when and how a notification appears. For example, Ubuntu uses the NotifyOSD daemon.

CHAPTER 1

# License and Contributors

notify2 is under the BSD 2-Clause License.

Some of the examples (icon.py, default-action.py, multi-actions.py and qt-app.py) are derived from pynotify examples, and are therefore LGPL-2.1, © 2006 Christian Hammond <chipx86@chipx86.com>.

## Contributors

- Thomas Kluyver
- John Terry

## License text

Copyright (c) 2012, Thomas Kluyver & contributors

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED

TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

notify2.**init**(*app_name*, *mainloop=None*)

Initialise the D-Bus connection. Must be called before you send any notifications, or retrieve server info or capabilities.

To get callbacks from notifications, DBus must be integrated with a mainloop. There are three ways to achieve this:

- Set a default mainloop (dbus.set_default_main_loop) before calling init()
- Pass the mainloop parameter as a string 'glib' or 'qt' to integrate with those mainloops. (N.B. passing 'qt' currently makes that the default dbus mainloop, because that's the only way it seems to work.)
- Pass the mainloop parameter a DBus compatible mainloop instance, such as dbus.mainloop.glib.DBusGMainLoop().

If you only want to display notifications, without receiving information back from them, you can safely omit mainloop.

notify2.**get_server_caps**()

Get a list of server capabilities.

These are short strings, listed in the spec. Vendors may also list extra capabilities with an 'x-' prefix, e.g. 'x-canonical-append'.

notify2.**get_server_info**()

Get basic information about the server.

# CHAPTER 2

## Creating and showing notifications

**class** notify2.**Notification**(*summary*, *message=''*, *icon=''*)
A notification object.

**summary** [str] The title text

**message** [str] The body text, if the server has the 'body' capability.

**icon** [str] Path to an icon image, or the name of a stock icon. Stock icons available in Ubuntu are listed here. You can also set an icon from data in your application - see *set_icon_from_pixbuf()*.

**show**()
Ask the server to show the notification.

Call this after you have finished setting any parameters of the notification that you want.

**update**(*summary*, *message=''*, *icon=None*)
Replace the summary and body of the notification, and optionally its icon. You should call *show()* again after this to display the updated notification.

**close**()
Ask the server to close this notification.

# CHAPTER 3

## Extra parameters

**class** notify2.**Notification**

**set_urgency**(*level*)
Set the urgency level to one of URGENCY_LOW, URGENCY_NORMAL or URGENCY_CRITICAL.

**set_timeout**(*timeout*)
Set the display duration in milliseconds, or one of the special values EXPIRES_DEFAULT or EXPIRES_NEVER. This is a request, which the server might ignore.

Only exists for compatibility with pynotify; you can simply set:

```
n.timeout = 5000
```

**set_category**(*category*)
Set the 'category' hint for this notification.

See categories in the spec.

**set_location**(*x*, *y*)
Set the notification location as (x, y), if the server supports it.

**set_icon_from_pixbuf**(*icon*)
Set a custom icon from a GdkPixbuf.

**set_hint**(*key*, *value*)
n.set_hint(key, value) <–> n.hints[key] = value

See hints in the spec.

Only exists for compatibility with pynotify.

**set_hint_byte**(*key*, *value*)
Set a hint with a dbus byte value. The input value can be an integer or a bytes string of length 1.

CHAPTER 4

# Callbacks

To receive callbacks, you must have set a D-Bus event loop when you called *init()*.

**class** notify2.**Notification**

**connect**(*event*, *callback*)
Set the callback for the notification closing; the only valid value for event is 'closed' (the parameter is kept for compatibility with pynotify).

The callback will be called with the *Notification* instance.

**add_action**(*action*, *label*, *callback*, *user_data=None*)
Add an action to the notification.

Check for the 'actions' server capability before using this.

**action** [str] A brief key.

**label** [str] The text displayed on the action button

**callback** [callable] A function taking at 2-3 parameters: the Notification object, the action key and (if specified) the user_data.

**user_data :** An extra argument to pass to the callback.

CHAPTER 5

# Constants

notify2.**URGENCY_LOW**
notify2.**URGENCY_NORMAL**
notify2.**URGENCY_CRITICAL**
: Urgency levels to pass to *Notification.set_urgency()*.

notify2.**EXPIRES_DEFAULT**
notify2.**EXPIRES_NEVER**
: Special expiration times to pass to *Notification.set_timeout()*.

CHAPTER 6

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## n

# Index