Desktop Notifications Specification

Mike Hearn

mike@navi.cx

Christian Hammond

chipx86@chipx86.com

Table of Contents

Introduction

This is a draft standard for a desktop notifications service, through which applications can generate passive popups (sometimes known as "poptarts") to notify the user in an asynchronous manner of events.

This specification explicitly does not include other types of notification presentation such as modal message boxes, window manager decorations or window list annotations.

Example use cases include:

- Presence changes in IM programs: for instance, MSN Messenger on Windows pioneered the use of passive popups to indicate presence changes.
- Scheduled alarm
- Completed file transfer
- New mail notification
- Low disk space/battery warnings

Basic Design

In order to ensure that multiple notifications can easily be displayed at once, and to provide a convenient implementation, all notifications are controlled by a single session-scoped service which exposes a D-BUS interface.

On startup, a conforming implementation should take the `org.freedesktop.Notifications` service on the session bus. This service will be referred to as the "notification server" or just "the server" in this document. It can optionally be activated automatically by the bus process, however this is not required and notification server clients must not assume that it is available.

The server should implement the `org.freedesktop.Notifications` interface on an object with the path `"/org/freedesktop/Notifications"`. This is the only interface required by this version of the specification.

A notification has the following components:

Table 1. Notification Components

Component	Description
Application Name	This is the optional name of the application sending the notification. This should be the application's formal name, rather than some sort of ID. An example would be "FredApp E-Mail Client," rather than "fredapp-email-client."
Application Icon	The application icon. This is represented either as a path or a name in an icon theme.
Replaces ID	An optional ID of an existing notification that this notification is intended to replace.
Notification Type ID	An optional ID representing the notification type. See Notification Types.

Component	Description
Urgency Level	The urgency of the notification. See Urgency Levels.
Summary	This is a single line overview of the notification. For instance, "You have mail" or "A friend has come online". It should generally not be longer than 40 characters, though this is not a requirement, and server implementations should word wrap if necessary. The summary must be encoded using UTF-8.
Body	This is a multi-line body of text. Each line is a paragraph, server implementations are free to word wrap them as they see fit. The text may contain simple markup as specified in Markup. It must be encoded using UTF-8. If the body is omitted just the summary is displayed.
Images	See Icons.
Actions	The actions send a request message back to the notification client when invoked. This functionality may not be implemented by the notification server, conforming clients should check if it is available before using it (see the GetCapabilities message in Protocol. An implementation is free to ignore any requested by the client. As an example one possible rendering of actions would be as buttons in the notification popup.
Hints	See Hints.
Expiration Time	The timestamp in seconds since the epoch that the notification should close. For example, if one wishes to have an expiration of 5 seconds from now, they must grab the current timestamp and add 5 seconds to it. If zero, the notification's expiration time is dependent on the notification server's settings, and may vary for the type of notification.

Each notification displayed is allocated a unique ID by the server. This is unique within the session. While the notification server is running, the ID will not be recycled unless the capacity of a uint32 is exceeded.

This can be used to hide the notification before the expiration time is reached. It can also be used to atomically replace the notification with another. This allows you to (for instance) modify the contents of a notification while it's on-screen.

Backwards Compatibility

Clients should try and avoid making assumptions about the presentation and abilities of the notification server. The message content is the most important thing.

Clients can check with the server what capabilities are supported using the `GetCapabilities` message. See Protocol.

If a client requires a response from a passive popup, it should be coded such that a non-focus-stealing message box can be used in the case that the notification server does not support this feature.

Markup

Body text may contain markup. The markup is XML-based, and consists of a small subset of HTML along with a few additional tags.

The following tags should be supported by the notification server. Though it is optional, it is recommended. Notification servers that do not support these tags should filter them out.

`<b>` ... `</b>`	Bold
`<i>` ... `</i>`	Italic
`<u>` ... `</u>`	Underline
`<a href="...">` ... `</a>`	Hyperlink

* *What else do we want here? We're going to want more tags for sure.*

Icons

A notification can optionally include an array of images. The array of images specifies frames in an animation, which always loop. Implementations are free to ignore the images data, and implementations that support images need not support animation.

If the image array has more than one element, a "primary frame" can be specified. If not specified, it defaults to the first frame. For implementations that support images but not animation, only the primary frame will be used.

Each element of the array must have the same type as the first element. Mixtures of strings and blobs are not allowed. The element types can be one of the following:

Element	Type	Description
Icon Theme Name	String	Any string that does not begin with the / character is assumed to be an icon theme name and is looked up according to the spec. The best size to fit the servers chosen presentation will be used. This is the recommended way of specifying images.

Element	Type	Description
Absolute Path	String	Any string that begins with a / will be used as an absolute file path. Implementations should support at minimum files of type image/png and image/svg.
Image Data	Binary Data	A data stream may be embedded in the message. This is assumed to be of type image/png.

Notification Types

Notifications can optionally have a type indicator. Although neither client or nor server must support this, some may choose to. Those servers implementing notification types may use them to intelligently display the notification in a certain way, or group notifications of similar types.

The following table lists standard notifications as defined by this spec. More will be added in time.

Table 2. Notification Types

Type	Description
`"email"`	An e-mail notification.
`"im"`	A new IM notification.
`"device"`	A device-related notification, such as a USB device being plugged in or unplugged.
`"presence"`	A presence change, such as a user going online or offline.
`"transfer-complete"`	A file transfer or download complete notification.

Third parties, when defining their own notification types, should discuss the possibility of standardizing on the hint with other parties, preferably in a place such as the xdg[1] mailing list at freedesktop.org[2]. If it warrants a standard, it will be added to the table above. If no consensus is reached, the notification type should be in the form of `"x-`*`vendor-name`*`."`

Urgency Levels

Notifications have an urgency level associated with them. This defines the importance of the notification. For example, "Your computer is on fire" would be a critical urgency. "Joe Bob signed on" would be a low urgency.

Urgency levels are defined as follows:

Table 3. Urgency Levels

Type	Description

Type	Description
0	Low
1	Medium (Normal)
2	High
3	Critical

Developers must use their own judgement when deciding the urgency of a notification. Typically, if the majority of programs are using the same level for a specific type of urgency, other applications should follow them.

For the most part, server implementations may use urgency information how they see fit. The one exception is the Critical notification. As Critical notifications are things that the user will most likely want to know about, they should not be closed until the user dismisses them.

Hints

Hints are a way to provide extra data to a notification server that the server may be able to make use of.

Neither clients nor notification servers are required to support any hints. Both sides should assume that hints are not passed, and should ignore any hints they do not understand.

Third parties, when defining their own hints, should discuss the possibility of standardizing on the hint with other parties, preferably in a place such as the xdg[3] mailing list at freedesktop.org[4]. If it warrants a standard, it will be added to the table above. If no consensus is reached, the hint name should be in the form of "`x-`*`vendor`*`-`*`name`*`.`"

D-BUS Protocol

The following messages *must* be supported by all implementations.

Message commands

`org.freedesktop.Notifications.GetCapabilities`

```
STRING_ARRAY org.freedesktop.Notifications.GetCapabilities (void);
```

This message takes no parameters.

It returns an array of strings. Each string describes an optional capability implemented by the server. The following values are defined by this spec:

Table 4. Server Capabilities

`"body"`	Supports body text. Some implementations may only show the summary (for instance, onscreen displays, marquee/scrollers)

"markup"	Supports markup in the body text. If marked up text is sent to a server that does not give this cap, the markup will show through as regular text so must be stripped clientside.
"static-image"	Supports display of exactly 1 frame of any given image array. This value is mutually exclusive with "multi-image", it is a protocol error for the server to specify both.
"multi-image"	The server will render an animation of all the frames in a given image array. The client may still specify multiple frames even if this cap and/or static-image is missing, however the server is free to ignore them and use only the primary frame.
"actions"	The server will provide the specified actions to the user. Even if this cap is missing, actions may still be specified by the client, however the server is free to ignore them.

New vendor-specific caps may be specified as long as they start with "x-*vendor*". For instance, "x-gnome-foo-cap". Capability names must not contain spaces. They are limited to alpha-numeric characters and dashes ("-").

org.freedesktop.Notifications.Notify

```
UINT32 org.freedesktop.Notifications.Notify (STRING_OR_NIL app_name,
BYTE_ARRAY_OR_STRING_OR_NIL app_icon, UINT32_OR_NIL replaces_id,
STRING_OR_NIL notification_type, BYTE urgency_level, STRING summary,
STRING_OR_NIL body, ARRAY images, DICT_OR_NIL actions, DICT_OR_NIL
hints, UINT32_OR_NIL expire_time);
```

Sends a notification to the notification server.

Table 5. Notify Parameters

Name	Type	Description
app_name	STRING or NIL	The optional name of the application sending the notification.
app_icon	BYTE_ARRAY or STRING or NIL	The optional program icon of the calling application. This is in the same format as an image frame. See Icons.

Name	Type	Description
replaces_id	UINT32 or NIL	The optional notification ID that this notification replaces. The server must atomically (ie with no flicker or other visual cues) replace the given notification with this one. This allows clients to effectively modify the notification while it's active.
notification_type	STRING or NIL	The optional notification type ID, for potential server categorization and logging purposes. See Notification Types.
urgency_level	BYTE	The urgency level. See Urgency Levels.
summary	STRING	The summary text briefly describing the notification.
body	STRING or NIL	The optional detailed body text.
images	ARRAY or NIL	The optional array of images. See Icons.
actions	DICT or NIL	A dictionary key of actions. Each key is the localized name of the action, as it should appear to the user, and maps to a UINT32 value containing a program-specific action code. This code will be reported back to the program if the action is invoked by the user.
hints	DICT or NIL	Optional hints that can be passed to the server from the client program. Although clients and servers should never assume each other supports any specific hints, they can be used to pass along information, such as the process PID or window ID, that the server may be able to make use of. See Hints.

Name	Type	Description
expire_time	UINT32 or NIL	The notification time-out time, represented as UNIX-time (seconds since the epoch). If this is NIL, the notification will never time out, and will only be closed when an action is invoked. If non-NIL, this will specify a time at which the notification will be automatically closed. If zero, the server's default expiration time will be used.

If *replaces_id* is NIL, the return value is a UINT32 that represent the notification. It is unique, and will not be reused unless a `MAXINT` number of notifications have been generated. An acceptable implementation may just use an incrementing counter for the ID. The returned ID is always greater than zero. Servers must make sure not to return zero as an ID.

If *replaces_id* is not NIL, the returned value is the same value as *replaces_id*.

org.freedesktop.Notifications.CloseNotification

```
void org.freedesktop.Notifications.CloseNotification (UINT32 id);
```

Causes a notification to be forcefully closed and removed from the user's view. It can be used, for example, in the event that what the notification pertains to is no longer relevant, or to cancel a notification with no expiration time.

The `NotificationClosed` signal is emitted by this method.

If the notification no longer exists, an empty D-BUS Error message is sent back.

org.freedesktop.Notifications.GetServerInformation

```
void org.freedesktop.Notifications.GetServerInformation (out STRING name,
out STRING vendor, out STRING version);
```

This message returns the information on the server. Specifically, the server name, vendor, and version number.

Table 6. GetServerInformation Return Values

Name	Type	Description
name	STRING	The product name of the server.

Name	Type	Description
vendor	STRING	The vendor name. For example, "KDE," "GNOME," "freedesktop.org," or "Microsoft."
version	STRING	The server's version number.

Signals

org.freedesktop.Notifications.NotificationClosed

```
org.freedesktop.Notifications.NotificationClosed (UINT32 id, UINT32
reason);
```

A completed notification is one that has timed out, or has been dismissed by the user.

Table 7. NotificationClosed Parameters

Name	Type	Description
id	UINT32	The ID of the notification that was closed.
reason	UINT32	The reason the notification was closed. 1 - The notification expired. 2 - The notification was dismissed by the user. 3 - The notification was closed by a call to `CloseNotification`. 4 - Undefined/reserved reasons.

The ID specified in the signal is invalidated *before* the signal is sent and may not be used in any further communications with the server.

org.freedesktop.Notifications.ActionInvoked

```
org.freedesktop.Notifications.ActionInvoked (UINT32 id, UINT32
action_id);
```

This signal is emitted when one of the following occurs:

- The user performs some global "invoking" action upon a notification. For instance, clicking somewhere on the notification itself.
- The user invokes a specific action as specified in the original Notify request. For example, clicking on an action button.

Table 8. ActionInvoked Parameters

Name	Type	Description
id	UINT32	The ID of the notification emitting the ActionInvoked signal.
action_id	UINT32	The ID of the action invoked. A value of 0 means that the default action was invoked, i.e., clicking the notification itself. IDs greater than zero are the action IDs as defined by the calling application.

Note: Clients should not assume the server will generate this signal. Some servers may not support user interaction at all, or may not support the concept of being able to "invoke" a notification.

Notes

1. http://freedesktop.org/mailman/listinfo/xdg
2. http://freedesktop.org/
3. http://freedesktop.org/mailman/listinfo/xdg
4. http://freedesktop.org/