System Tray Protocol Specification

Havoc Pennington

<hp@redhat.com>

Version 0.2

Table of Contents

Overview

The "system tray" is an application running on a given X screen that can display small icons provided by running applications. Windows XP calls this feature the "notification area." [1] Inspired by KDE, this specification uses the term "system tray."

From a UI standpoint, the system tray is normally used for transient icons that indicate some special state, while full-blown "applets" are used for permanent dock/panel features. For example, a system tray icon might appear to tell the user that they have new mail, or have an incoming instant message, or something along those lines.

The basic idea is that creating an icon in the notification area is less annoying than popping up a dialog. However it's also harder to notice, so Windows XP adds a feature allowing tray icons to pop up small message balloons. (Users can disable these via a hidden registry setting.) This specification also supports the balloon feature.

Definitions

System tray

> The system tray is an X client which owns a special manager selection on a given screen and provides container windows.

Selection owner window

> The selection owner window is the window belonging to the System Tray that owns the manager selection (as in `XGetSelectionOwner()`/`XSetSelectionOwner()`. Note that this probably is not the same window that's used to contain the system tray icons.

Tray icon

> The tray icon is a window to be embedded in the system tray.

Locating the system tray

On startup, the system tray must acquire a manager selection called `_NET_SYSTEM_TRAY_Sn`, replacing `n` with the screen number the tray wants to use. The conventions for manager selections are defined in the ICCCM.

Because the selection owner window should be destroyed when the manager selection is lost, normally the selection owner window will not be the same as any of the user-visible windows provided by the system tray.

A system tray that fails to get the selection or loses the selection should assume that another system tray is running, and let the selection owner handle tray icons.

An application wishing to provide an icon to the system tray should first locate the system tray by requesting the owner window of the manager selection. If the manager selection has no owner, clients may use the method described in the ICCCM (watching for a `MANAGER` client message) to be notified when a system tray appears.

Opcode messages

Tray icons can send "opcodes" to the system tray. These are X client messages, sent with `NoEventMask`, a `message_type` of `_NET_SYSTEM_TRAY_OPCODE`, and format 32. The first data field in the message is a timestamp (the stamp of the current event, if available, otherwise CurrentTime). The second data field is an integer indicating the op code of the message:

```
#define SYSTEM_TRAY_REQUEST_DOCK    0
#define SYSTEM_TRAY_BEGIN_MESSAGE   1
#define SYSTEM_TRAY_CANCEL_MESSAGE  2
```

The content remaining three data fields depends on the type of message being sent. If they are unused by a particular message, they should always be set to 0.

Here is an example of how to send a client message:

```
#include <X11/Xlib.h>

void send_message(
     Display* dpy, /* display */
     Window w,     /* sender (tray icon window) */
     long message, /* message opcode */
     long data1    /* message data 1 */
     long data2    /* message data 2 */
     long data3    /* message data 3 */
){
    XEvent ev;

    memset(&ev, 0, sizeof(ev));
    ev.xclient.type = ClientMessage;
    ev.xclient.window = w;
    ev.xclient.message_type = XInternAtom (dpy, "_NET_SYSTEM_TRAY_OPCODE", False );
    ev.xclient.format = 32;
    ev.xclient.data.l[0] = x_time;
    ev.xclient.data.l[1] = message;
    ev.xclient.data.l[2] = data1;
    ev.xclient.data.l[3] = data2;
    ev.xclient.data.l[4] = data3;

    trap_errors();
    XSendEvent(dpy, w, False, NoEventMask, &ev);
    XSync(dpy, False);
    if (untrap_errors()) {
        /* Handle failure */
    }
}
```

Docking a tray icon

A tray icon must support the "client" or "plug" side of the XEMBED specification. XEMBED is a protocol for cross-toolkit widget embedding.

To begin the docking process, the tray icon application sends a client message event to the manager selection owner window, as described in the section called “Opcode messages”. This event should contain the `SYSTEM_TRAY_REQUEST_DOCK` opcode, `xclient.data.l[2]` should contain the X window ID of the tray icon to be docked.

At this point the "embedding life cycle" explained in the XEMBED specification begins. The XEMBED specification explains how the embedding application will interact with the embedded tray icon, and how the embedder/embedded relationship may be ended.

Tray icons may be assigned any size by the system tray, and should do their best to cope with any size effectively.

Tray icon hints

Tray icons should set the following hints to help the system tray provide a nice user interface. The name and icon hints are used if the system tray needs to refer to a tray icon; for example, the system tray may present a list of tray icons and let the user reorder them or change their properties.

_NET_WM_NAME

```
_NET_WM_NAME, UTF8_STRING
```

This hint should be set as it would be for a normal toplevel window, as defined in the Extended Window Manager Hints Specification (EWMH). The hint MUST be in UTF-8 encoding. It provides a human-readable, localized name for the tray icon.

WM_CLASS

```
WM_CLASS, STRING
```

This hint should be set as it would be for a normal toplevel window, as defined in the ICCCM. The system tray can use it to distinguish different kinds of tray icon. This is useful for example if the system tray wants to save and restore the positions of the icons in the tray.

_NET_WM_ICON

```
_NET_WM_ICON CARDINAL[][2+n]/32
```

This hint should be set as it would be for a normal toplevel window, as defined in the Extended Window Manager Hints Specification (EWMH). See that specification for the format of the icon data.

Tray manager hints

The tray manager should set the following hints on the selection owner window.

_NET_SYSTEM_TRAY_ORIENTATION

```
_NET_SYSTEM_TRAY_ORIENTATION orientation CARDINAL/32

  #define _NET_SYSTEM_TRAY_ORIENTATION_HORZ 0
  #define _NET_SYSTEM_TRAY_ORIENTATION_VERT 1
```

The property should be set by the tray manager to indicate the current orientation of the tray. Tray icons may use this hint in order to maintain the icon's aspect ratio and also as an indication of how the icon contents should be laid out.

Balloon messages

Tray icons may ask the system tray to display a balloon message to the user. The system tray coordinates balloon messages to ensure that they have a consistent look-and-feel, and to avoid displaying multiple balloon

messages at once.

A balloon message is a short text message to be displayed to the user. The message may have a timeout; if so, the message will be taken down after the timeout expires. Messages are displayed in a queue, as only one can appear at a time; if a message has a timeout, the timer begins when the message is first displayed. Users may be allowed to close messages at any time, and may be allowed to disable all message display.

System trays may display balloon messages in any way they see fit; for example, instead of popping up a balloon, they could choose to put a special indicator around icons with pending messages, and display the message on mouseover.

Balloon messages are sent from the tray icon to the system tray selection owner window as a series of client messages. The first client message is an opcode message, and contains the usual timestamp, and the op code `SYSTEM_TRAY_BEGIN_MESSAGE`. `xclient.data.l[2]` contains the timeout in thousandths of a second or zero for infinite timeout, `xclient.data.l[3]` contains the length of the message string in bytes, not including any nul bytes, and `xclient.data.l[4]` contains an ID number for the message. This ID number should never be reused by the same tray icon. (The simplest way to generate the ID number is to increment it with each message sent.)

Following the `SYSTEM_TRAY_BEGIN_MESSAGE` op code, the tray icon should send a series of client messages with a `message_type` of `_NET_SYSTEM_TRAY_MESSAGE_DATA`. These client messages must have their `window` field set to the window ID of the tray icon, and have a `format` of 8.

Each `_NET_SYSTEM_TRAY_MESSAGE_DATA` message contains 20 bytes of the message string, up to the length given in the `SYSTEM_TRAY_BEGIN_MESSAGE` opcode. If the message string is zero-length, then no messages need be sent beyond the `SYSTEM_TRAY_BEGIN_MESSAGE`. A terminating nul byte should never be sent.

System trays may receive portions of messages from several tray icons at once, so are required to reassemble the messages based on the window ID of the tray icon.

The tray icon may wish to cancel a previously-sent balloon message. To do so, it sends a `SYSTEM_TRAY_CANCEL_MESSAGE` opcode with `data.l[2]` set to the ID number of the message to cancel.

A. Change history

Version 0.2, 23 November 2004, Mark McLoughlin.

- Added the _NET_SYSTEM_TRAY_ORIENTATION hint.

Version 0.1, 20 April 2002, Havoc Pennington.

- Created initial draft.

---

[1] According to the MSDN documentation for the `Shell_NotifyIcon()` function, "The taskbar notification

area is sometimes erroneously called the 'tray.'" So presumably "notification area" is the official term on Windows. Parts of the docs also call it the "status area."