

Re: WM_COPYDATA

Source: <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2007-09/msg00013.html>

- *From:* "Maarten Wiltink" <maarten@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sun, 9 Sep 2007 15:05:07 +0200
-

<jimbo@xxxxxxxx> wrote in message
news:1189327537.781216.122260@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

On Sep 7, 12:09 pm, "Maarten Wiltink" <maar...@xxxxxxxxxxxxxxxxxxxxx>
wrote:

<WM_COPYDATA>

For a stringlist, send its Text property. That's all the lines concatenated together, with end-of-line sequences (CR LF) to separate them. On the receiving end, assign the text to the Text property of the receiving TStrings object and it will split it into lines again.

[...]

For an array of strings, you can do the same thing, only you have to write the code yourself.

Actually, I have a stringlist with objects. I feared to go into detail in the op.

Excellent. Restricting the first question to strings is going to work out well.

[...] will the code run synch? Will the receiver process the data before the sender frees it?

That depends. If you use SendMessage, the window procedure of the receiver is called synchronously. With PostMessage, the message is queued and the PostMessage call returns immediately.

Re: WM_COPYDATA

The trick, in the second case, is to send data that don't reference anything that may be freed.

How would I cast/send mypiclist in order to receive and process the objects if need be?

The normal solution is to serialise your objects and send not the object, not a reference to it, but data that allow a copy to be reconstituted at the other end. This is also the limitation: if you need changes at the other end to be visible at the original end, it doesn't happen automatically. Of course, if you can correlate the objects on both ends, you can stream updates back and forth.

This is a big subject.

If your problem is not necessarily with the objects that denote things (you were talking about a selection, I think), but with the data they encapsulate, it's not a big problem. Realise that you have to extract the data from your objects on one side, and process it on the other side. In between, forget about the objects, and write yourself a protocol.

Protocols are a much smaller subject.

The protocol should focus on the problem domain, not the objects with which you implement it. It should be possible to use different objects on both sides.

As an example, let me describe a possible (not the definitive) implementation for pushing images to a server (Receiver). The client (Sender) should be able to send messages to create a new image, delete it, and fill in the picture. Only selected images are pushed, and the actual picture is sent rather than a filename.

It's a text-based protocol (because then you can write a server and test it by Telnetting to it, without first having to write a working client as well), and it identifies objects by codes so you can send several messages regarding the same object.

To create an object: CREATE <code> <width> <height> CRLF
To destroy an object: DESTROY <code> CRLF
To paint an object: LINE <code> <row> RGBA RGBA RGBA ... <CRLF>

To send a glyph for the character 'A' (in 10pt Courier New):

```
CREATE 65 9 9  
LINE 65 0 FFF0 FFF0 FFF0 FFF0 FFF0 FFF0 FFF0 FFF0 FFF0  
LINE 65 1 FFF0 FFF0 FFF0 0000 0000 FFF0 FFF0 FFF0 FFF0
```

Re: WM_COPYDATA

Re: WM_COPYDATA

```
LINE 65 2 FFF0 FFF0 FFF0 FFF0 0000 FFF0 FFF0 FFF0 FFF0
LINE 65 3 FFF0 FFF0 FFF0 0000 FFF0 0000 FFF0 FFF0 FFF0
LINE 65 4 FFF0 FFF0 FFF0 0000 FFF0 0000 FFF0 FFF0 FFF0
LINE 65 5 FFF0 FFF0 FFF0 0000 0000 0000 FFF0 FFF0 FFF0
LINE 65 6 FFF0 FFF0 0000 FFF0 FFF0 FFF0 0000 FFF0 FFF0
LINE 65 7 FFF0 0000 0000 0000 FFF0 0000 0000 0000 FFF0
LINE 65 8 FFF0 FFF0 FFF0 FFF0 FFF0 FFF0 FFF0 FFF0 FFF0
```

I used 4 bits per colour channel to keep the lines shorter, but that doesn't really matter. Your protocol parser could even be flexible about it. (It should probably also be flexible about too short or too long lines, duplicate objects, and messages to nonexistent objects.)

Note that this is not a request-reply protocol but one-way. There is no feedback if anything goes wrong, but it is somewhat robust against errors. You can always wait for the next CRLF if you lose track, and you can always ignore messages to objects that don't exist.

Writing protocol handlers can be quite a bit of work, but it very nicely decouples things.

Groetjes,
Maarten Wiltink