

# Re: WM\_COPYDATA

---

*Source:* <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2007-09/msg00020.html>

---

- *From:* "Maarten Wiltink" <[maarten@xxxxxxxxxxxxxxxxxxxxx](mailto:maarten@xxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Mon, 10 Sep 2007 11:20:55 +0200
- 

<jimbo@xxxxxxxxxx> wrote in message  
[news:1189405714.732896.83960@xx](mailto:news:1189405714.732896.83960@xx)

On Sep 9, 4:05 pm, "Maarten Wiltink" <[maar...@xxxxxxxxxxxxxxxxxxxxx](mailto:maar...@xxxxxxxxxxxxxxxxxxxxx)>  
wrote:

[...]

The trick, <with PostMessage>, is to send data that don't reference anything that may be freed.

Yes. I did some homework and testing with both. I had trouble getting the type casting correct.

That's always going to be a problem with sending messages. The mechanism is quite general (and minimal), and to it, everything is an integer. But you seem to be doing well so far.

[...] Here is the solution.

[...]

```
for i := 0 to mypiclist.Count - 1 do
if tpictureobj(mypiclist[i]).Selected then
begin
picobj := tpictureobj(mypiclist.Objects[i]);
mylist.Add(picobj.PicPath);
End;
```

Looks good.

The normal solution is to serialise your objects and send not the

Re: WM\_COPYDATA

object, not a reference to it, but data that allow a copy to be reconstituted at the other end.

Similar to the SendImage example in Gajics code?

Very probably, yes. I haven't seen the code but serialisation is a well-known technique, and well-suited for sending images. The important bit is to reconstruct the picture. The object involved don't really matter.

[...] working with objects and pointers doesn't bother me. It's not yet clear to me how Delphi handles them. I considered sending the actual images. One at a time is no problem but I haven't a clue how to serialize a set of images or how to handle them. I don't think it's necessary in my situation.

Delphi does not handle objects and pointers much at all. \*You\*, the programmer, are responsible for allocating and freeing all your data. Object variables are references, pointers. That's the main thing to keep in mind. Passing an object to a function passes it by reference.

'Sending the actual images' is not a very clear description. Sending object references is a bad idea; the receiving end probably has its own address space with the object simply not in it.

Sending a serialisation of the images is okay. Sending a filename to load it from will probably be just as good as sending pixel data, if there was a file to begin with.

I have one more problem with making this work. I need to initiate the original contact from the receiver. e.g. when receiver is run or button clicked it should ask the main app to send the data. Should I post this in a new thread or somewhere else?

Nothing wrong with this one.

Request-reply cycles are not at all problematic. One side sends a message, the other sends a message back. To agree on what message identifier to use, look up RegisterWindowMessage.

A convenient (but not very secure) way to avoid the mutual waiting which may get in the way of responding to other messages is to use a stateless

Re: WM\_COPYDATA

Re: WM\_COPYDATA

protocol, where requests are simply sent and the client is always ready to receive replies. Effectively, there are two simplex client-server channels in opposite directions.

You open yourself to spoofing/poisoning attacks, but you don't need any logic to relate responses to requests, or timeouts, or threads or other horrible things.

[...]

Protocols are a much smaller subject.

Maybe for you ;>)

You could understand this one, right? Protocols need not be complex; in fact, it's better if they aren't. There are several simple tricks to make life easy. Make them text-based, stateless if you can, with orthogonal commands. Make each command simple to parse and simple to execute; adding commands is much easier than doing extra work for existing commands.

[...]

[...] I imagine it will be some kind of broadcast and registermessage thing. Windows messaging terrifies me.

Ah, you already knew of Register(Window)Message. Windows messaging is actually really easy. It's so limited that you can't do much wrong. Everything you want to do, you have to fit into one or two integers. When the building blocks are that small, it's almost impossible to glue them together in a way that doesn't work.

And while you call yourself new at Delphi, you seem to have a few good habits that betray enough experience with programming in general to get through this. You'll learn to love Delphi before long. (-:

Groetjes,  
Maarten Wiltink

.

Re: WM\_COPYDATA