

## Re: CString over LPARM?

**Source:** <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2004-02/0721.html>

---

**From:** Joseph M. Newcomer (*newcomer\_at\_flounder.com*)

**Date:** 02/16/04

Date: Mon, 16 Feb 2004 02:06:19 -0500

You are probably wrong. If you use a char buffer that is a local variable, it will likely be gone by the time the message is processed. If you use one which is a class member, static, or global (the latter two being incredibly poor choices), then you have the problem of what happens if you post two messages.

For example

```
function()
{
    char buffer[100];
    strcpy(buffer, "this is a test");
    wnd->PostMessage(UWM_WHATEVER, (WPARAM)buffer);
}
```

will produce something between the string you give, complete nonsense, and an access fault when you try to use the value. In rare and unusual circumstances, you might see the actual value. Most of the time you will get trash, and sometimes you will generate access faults. This is because the buffer is gone when the message is processed.

If you do

```
function(CString whatever)
{
    static char buffer[100];
    strcpy(buffer, whatever);
    wnd->PostMessage(UWM_WHATEVER, (WPARAM)buffer);
}
```

means the buffer address will be valid, but the contents will be trash; imagine calling the function twice, once with the string "this" and once with the string "that". The receiving thread of the post will most likely display "that" twice, because the address points to the same place for both messages but the contents have been trashed.

The only safe solution I know of is to allocate a heap object for each request and delete it when the message is received.

```
CString * s = new CString;
s->Format("%s %d", "The value is", n); // silly example, but shows the point
```

microsoft.public.vc.mfc: Re: CString over LPARM?

```
wnd->PostMessage(UWM_WHATEVER, (WPARAM)s);
```

and the receiving function would be

```
LRESULT classname::OnWhatever(WPARAM wParam, LPARAM)
{
    CString * s = (CString *)wParam;
    ... do something with CString value
    delete s; // unless you have stored the pointer for some other reason
    return 0;
}
```

joe

On Fri, 13 Feb 2004 16:59:27 +0100, "Holger Kreißl" <hok@informatik.tuchernitz.de> wrote:

>*I got it.*

>

>*I only have to use a char buffer instead of CString an it works.*

>

>

>*Holger*

>

Joseph M. Newcomer [MVP]

email: [newcomer@flounder.com](mailto:newcomer@flounder.com)

Web: <http://www.flounder.com>

MVP Tips: [http://www.flounder.com/mvp\\_tips.htm](http://www.flounder.com/mvp_tips.htm)