

Re: PInvoke Marshalling....

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-10/msg03078.html>

- *From:* "Nicholas Paldino [.NET/C# MVP]" <mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 23 Oct 2007 13:00:13 -0400
-

Daniel,

Can you provide a declaration for the structure in C#? Or C++? If you have a structure declaration in C#, you can call the static `PtrToStructure` method on the `Marshal` class to marshal the values from the memory pointed to by the unmanaged pointer to managed code.

--

- Nicholas Paldino [.NET/C# MVP]
- mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

"Daniel Bass" <danREMOVEbass@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:OzWx9OZFIHA.4748@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

That's excellent! Thanks for your help.

One last question...

The `wParam` is a pointer to a API defined structure (containing a `Low DWORD` and a `High DWORD`), how would I cast this back into the same structure I've written in my C# proxy in the `WndProc`?

"Nicholas Paldino [.NET/C# MVP]" <mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:eWDBtYFIHA.5160@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Daniel,

My previous post outlines what you have to do with allocating the buffer.

When you pass a `StringBuilder`, the P/Invoke layer creates a buffer of unmanaged memory, and passes the pointer to that, populated with the contents of the string builder. Upon return, the P/Invoke layer marshals the information from that unmanaged buffer back into the `StringBuilder`, and disposes of the pointer.

Re: PInvoke Marshalling....

However, this API is holding onto the pointer, so you have to do the same.

--

- Nicholas Paldino [.NET/C# MVP]
- mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

"Daniel Bass" <danREMOVEbass@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message
news:O78SjhYFIHA.4228@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

Well, kind of...

A little context may help. I'm writing a front end into a driver for a USB connected imaging scanner.

DeviceHandle is the handle that I keep hold of for the currently connected device.

The first call I make into the API though, is
[DllImport("Dependencies\\SNAPI.DLL")]
public static extern int SNAPI_Init(IntPtr hwend, IntPtr[] DeviceHandles, ref int NumDevices);
This basically expects a windows handler, along with a buffer array which it uses to populate with scanners that are currently attached to the system.

I've overridden WndProc, and it's correctly getting the message I'd expect, in this case it's the WM_VERSION message. All I'm trying to do is initially query the scanner for the firmware version as a simple way of exchanging data before I get onto imaging and video!

OnConnect type event does this:
int status = 0;
status = DriverProxy.SNAPI_SetVersionBuffer(handle, _versionBuffer, _versionBuffer.Length);
status = DriverProxy.SNAPI_TransmitVersion(handle);
_versionBuffer is global StringBuilder, and transmit version tells the scan to please send me the firmware version to the allocated buffer.

In WndProc override:
protected override void WndProc(ref Message m)

Re: PInvoke Marshalling....

```
{
switch ((uint)m.Msg)
{
case DriverProxy.WM_SWVERSION:
// should now have a _versionbuffer full of grand
stuff
FirmwareVersionLabel.Text =
_versionBuffer.ToString();
break;

}
Debug.WriteLineIf(m.Msg > 0x8000 && m.Msg < 0xBFFF,
"HIT! " +
m.Msg.ToString());
base.WndProc(ref m);
}
```

Now I've a label that gets the Firmware version text assigned to it, but this is currently blank, and can confirm that the WM_SWVERSION message comes through...

Hope that helps to explain what I'm trying to achieve.
Thanks again.
Dan.

"Nicholas Paldino [.NET/C# MVP]"
<mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote
in message
news:OadAbZYFIHA.3716@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

Daniel,

I was thinking this is a little odd, since the parameter is named "DeviceHandle" and not indicative that it is a windows handle. If it is indeed a windows handle you are passing to the method, then you can override the WndProc method of the control that contains that windows handle to look for the message sent to you.

You will have to change your API declaration to this:

```
[DllImport("Dependencies\\SNAPI.DLL",
CharSet=CharSet.Ansi)]
public static extern int
```

Re: PInvoke Marshalling....

```
SNAPI_SetVersionBuffer(IntPtr  
DeviceHandle,  
IntPtr pData, int max_length);
```

And then call it like this:

```
// dataBuffer need to be accessible by the  
// override of WndProc as well.  
dataBuffer =  
Marshal.AllocHGlobal(dataBufferLength);  
  
// Make the API call:  
SNAPI_SetVersionBuffer(windowHandle,  
dataBuffer, dataBufferLength);
```

Then, in your override of WndProc, in the section that handles the message to the window:

```
// Get the string.  
string data =  
Marshal.PtrToStringAnsi(dataBuffer);  
  
// Free the memory.  
Marshal.FreeHGlobal(dataBuffer);  
  
// Work with the string.
```

```
--  
- Nicholas Paldino [.NET/C# MVP]  
- mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
"Daniel Bass"  
<danREMOVEbass@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>  
wrote in message  
news:O3sK%23VYFIHA.3548@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

I pass in a handle of the windows into the API as part of an initialisation call. Could I then just override "WndProc"?

It's doing the latter, and only sending a message when the parameter has been sent to the buffer.

Re: PInvoke Marshalling....

When you say "allocate the memory" yourself, I'm not sure what you mean. I'll provide the IntPtr parameter, and according to the API manual, I'll have a global buffer space that I register with the API which should be populated with the event call... Does that sound feasible?

Thanks for your help and guidance!

"Nicholas Paldino [.NET/C# MVP]"

<mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>

wrote in message

news:OovEiMYFIHA.5208@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Daniel,

Yes, there is. You can create a class which implements the IMessageFilter interface and then call the AddMessageFilter method on the Application class, passing your implementation. Then, in the PreFilterMessage method, you would check for your message.

Re: PInvoke Marshalling....

There might be a more subtle issue here. When the function returns, is the string already written to? Or is the string written to when the message is sent to the application? If it is the former, then the code you have is ok. If it is the latter, you will have to change your declaration of the unsigned char * parameter to an IntPtr, and allocate the memory that you need yourself.

Then, in the handler for the message, you will have to manually marshal the string to managed code using the static methods on

Re: PInvoke Marshalling....

the
Marshal
class
(freeing the
memory of
course as
well).

--

- Nicholas
Paldino
[.NET/C#
MVP]

-

mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

"Daniel
Bass"

<danREMOVEbass@xxxxxxxxxxxxxxxxxxxxxxxx>

wrote in
message

news:u%23iPBKYFIHA.4140@xxxxxxxxxxxxxxxxxxxxxxxx

Nicholas,

I've
just
been
reading
through
the
API
guide,
and
realised
that
it
won't
pass
me
back
the
data
in
a
synchronous
manner.

To
quote:
"A

Re: Plnvoke Marshalling....

command
function
returns
immediately
to
the
host
application...
After
the
command
is
processed,
by
the
scanner,
the
host
application
received
a
Windows
message
indicating
the
command
was
processed.
The
host
applicaiton
provides
a
message
handler
for
the
ack
from
the
connected
device."

Eeek!
In
.Net,
is
there
a
way
to

Re: Plnvoke Marshalling....

Re: PInvoke Marshalling....

listen
for
messages?
I've
worked
a
little
with
message
maps,
but
that
was
way
back
and
have
not
see
this
sort
of
basic
message
handling
in
.net
before...

Cheers.
Dan.

"Nicholas
Paldino
[.NET/C#
MVP]"

<mvp@xx>

wrote
in
message

news:%23Wy5GFYFIHA.5328@xx

Daniel,

Yes,
if
the
API
function
is
going

Re: PInvoke Marshalling....

to
write
to
the
buffer,
then
passing
a
StringBuilder
will
cause
the
data
to
be
marshaled
back
to
you
correctly.

--

-

Nicholas
Paldino
[.NET/C#
MVP]

-

mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

"Daniel
Bass"

<danREMOVEbass@xxxxxxxxxxxxxxxxxxxxxxxx>

wrote

in

message

<news:%23EkQIBYFIHA.1212@xxxxxxxxxxxxxxxxxxxxxxxx>

Nicholas,

Excellent,
thanks
for
the
prompt
reply!

pData
is
a

Re: PInvoke Marshalling....

buffer
that
I
create,
then
pass
into
the
function
to
populate.
Does
the
parameter
as
a
StringBuilder
marshall
this
data
correctly?

Thanks.
Dan.

"Nicholas
Paldino
[.NET/C#
MVP]"
<mvp@xx>
wrote
in
message
news:ecx5T6XFIHA.280@xx

Daniel,

Actually,
it's
not
pretty
obvious.
In
C++,
long
corresponds
to
a
32-bit
integer,
which

Re: PInvoke Marshalling....

in
C#,
is
an
int.
Also,
you
should
declare
how
your
strings
are
marshaled
in
the
DllImport
attribute.
All-in-all,
your
declaration
should
look
like
this:

```
[DllImport("Dependencies\\SNAPI.D  
CharSet=CharSet.Ansi)]  
public  
static  
extern  
int  
SNAPI_SetVersionBuffer(IntPtr  
DeviceHandle,  
StringBuilder  
pData,  
int  
max_length);
```

Also,
if
the
pData
parameter
is
not
going
to
be
written
to

Re: PInvoke Marshalling....

```
*pData,  
long  
max_length);
```

```
So  
far  
I've  
got  
this:  
[DllImport("Dependencies\\S  
public  
static  
extern  
int  
SNAPI_SetVersionBuffer(In  
DeviceHandle,  
StringBuilder  
pData,  
long  
max_length);
```

```
Keep  
getting  
a  
PInvokeStackImbalance  
error  
and  
I  
think  
it's  
the  
2nd  
parameter  
pData  
because  
I've  
referenced  
the  
device  
handler  
and  
the  
long  
typedef  
is  
pretty  
obvious.
```

```
For  
the  
2nd
```

Re: PInvoke Marshalling....

paramter
I've
tried
the
following
without
success:
byte[]
char[]
string
StringBuilder
[MarshalAs(UnmanagedType)
string

HELP!
Thanks.
Dan.

Re: PInvoke Marshalling....