

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

July 2000 Index

[Main Page](#)[COL Archive](#)[Log In](#)[Subscribe](#)[Renewals](#)[Frequently Asked Questions](#)[Site Index](#)[Article Index](#)[Author Index](#)[Links To](#)[Other Sites](#)[Downloads](#)[Open Source](#)[Project](#)[Issues in](#)[PDF Format](#)[Free Software](#)[Advertising](#)[Contact Us](#)**TopSpeed****Clarion 5**
by TopSpeedFREE Microsoft
Internet Exploreretc2000
EVENT SPONSOR

[Web Builder Reporting](#)

One of the omissions in Web Builder, to date, appears to be reports. As of beta 2, for example, SoftVelocity has not decided how it is going to implement reports in Web Builder. This is quite a problem, or so it would appear.
(Jul 5,2000)

[The Clarion Advisor: EIP File Lookups](#)

Russ Eggen explains his technique for parsing file names out of the string returned by the EIP EditFileClass.
(Jul 5,2000)

[Please Mr. Postman: Calling External Functions By Address](#)

Steve Bottomley explains how to call an external procedure by address when the procedure isn't exported from the DLL.
(Jul 5,2000)

[Product Review: View Wizard 1.0 From Nice Touch Solutions](#)

Nice Touch Solutions' View Wizard adds user customization to your browses easily and quickly, and even lets users modify and save sort orders.
(Jul 11,2000)

[Sidebar Menus](#)

There have been a number of requests for a side-menu-bar-next-to-the-work-area style app frame. Steve Parker shows how it's done.
(Jul 11,2000)

[The Clarion Advisor: Memory Leaks And Virtual Methods](#)

Jeff Slarve finds and fixes a tricky memory leak.
(Jul 11,2000)

[Using API Threads - Part 1](#)

Clarion does an excellent job of managing threads for the developer, but there are times when the standard thread handling just doesn't cut it. Jim Kane shows how to safely use API threads in Clarion.
(Jul 18,2000)

[A Tale Of Three Brokers](#)

Steve Parker untangles the web of confusion around Clarion's three application brokers.
(Jul 18,2000)

[Clarion 5.5 Gold Candidate: A First Look](#)

Andrew Guidroz II takes a look at the upcoming C5.5 Gold candidate release.
(Jul 25,2000)

[Legacy to ABC: There is Another Way!](#)

Daunted by the challenge of migrating your apps from Legacy to ABC? Simon Brewer shows how to do it one piece at a time with a hybrid of Legacy and ABC code. Part 1 of a series.

(Jul 25,2000)

[Using API Threads - Part 2](#)

Clarion does an excellent job of managing threads for the developer, but there are times when the standard thread handling just doesn't cut it. Jim Kane shows how to safely use API threads in Clarion.

(Jul 25,2000)

[July 2000 News](#)

Clarion news, notes, and happenings from around the globe.

(Jul 25,2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source
Project](#)
[Issues in
PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)**TopSpeed****Clarion 5**
*by TopSpeed***FREE** Microsoft
**Internet
Explorer** **etc2000**
EVENT SPONSOR

Web Builder Reporting

by **Steve Parker**

One of the omissions in Web Builder, to date, *appears* to be reports. As of beta 2, for example, SoftVelocity has not decided how it is going to implement reports in Web Builder. This is quite a problem, or so it would appear.

"'Appear?'" Parker, with all due respect, are you out of your mind? Reports are a great gaping hole," some would claim and, indeed, have claimed, for both Internet Connect and Web Builder.

Since you're either calling me names or questioning (what's left of) my sanity, no, I don't think I've missed the boat here. Neither do I consider "missing" reports an issue of any significance.

If reports were a problem, how do you think I've been providing reports in my knowledge base? If anything calls for a report, presenting a FAQ article for reading does. For example, have a look at Figure 1.

Figure 1: Output when reading a FAQ article

[Web Builder Reporting](#)
(Jul 5,2000)[The Clarion Advisor: EIP
File Lookups](#)
(Jul 5,2000)[Please Mr. Postman:
Calling External Functions
By Address](#)
(Jul 5,2000)[July 2000 News](#)
(Jul 5,2000)

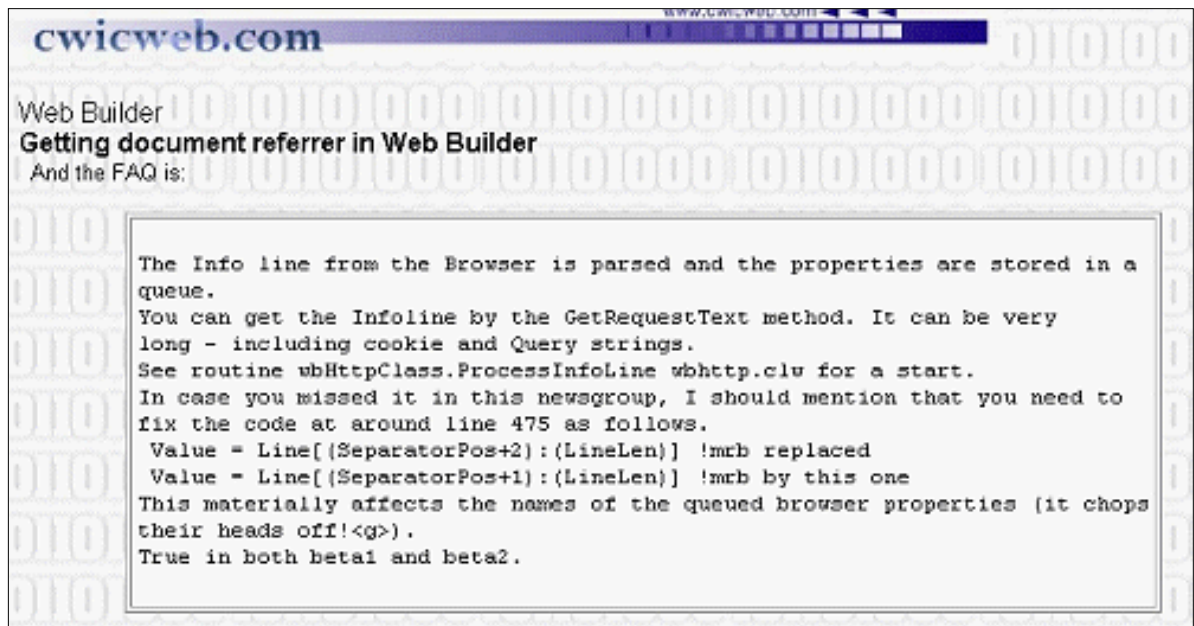


Figure 1 certainly *looks* like a report (or, at least, what a report looks like on the Web).

Similarly, I have been providing "print outs" of user selected jobs in my student job viewer apps (www.par2.com/cws/c5launch.dll/samn.exe.0 – which *really* looks like a report, in the most traditional sense – and www.par2.com/cws/c5launch.dll/samn2.exe.0) for several years.

Figure 2. Job report output.

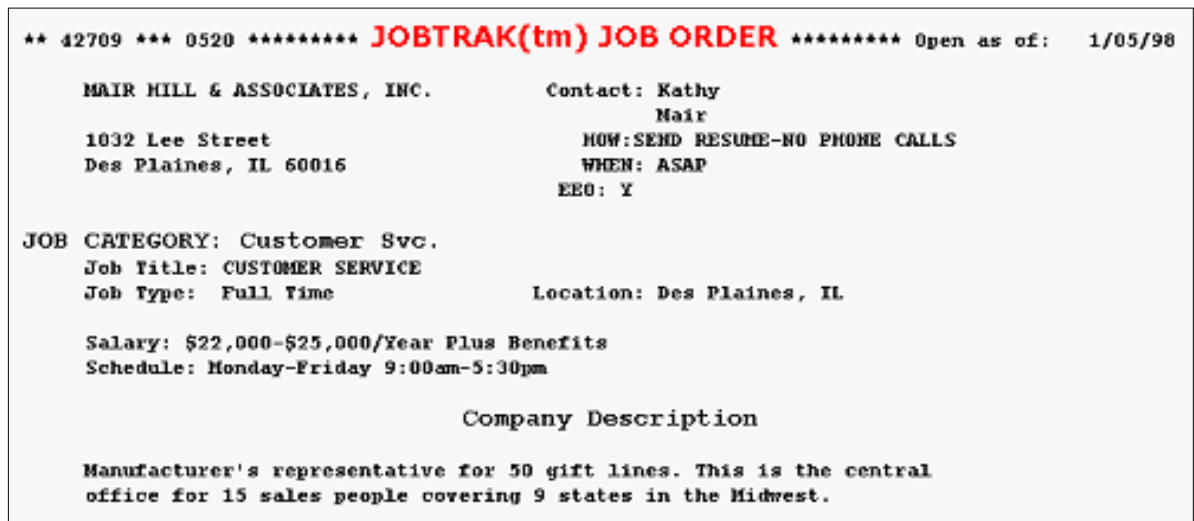


Figure 3. Internet Connect (Java) standard report output.

Page 1/27		X	
53,165	0504	JOBTRAK(tm) Job Order	Updated: 3/31/1997
AMERICAN EXPRESS FINANCIAL ADVISORS		CONTACT:	Laura Golembieski
270 E. 90th Dr.		HOW:	SEND RESUME
Merrillville, IN 46410		WHEN:	ASAP
219-769-0810		X FAX	# Openings: 5
JOB CATEGORY: Acctg & Finance			
Job Title: PERSONAL FINANCIAL ADVISOR			
Job Type: Full Time		Location: Chicagoland & NW Indiana	
Salary: \$24,000/Year Plus Commission - Unlimited Earnings		Starts: ASAP	
Schedule: Schedule To Be Determined			
Company Description:			
Financial planning and investment advisory services to individuals and businesses.			
Qualifications/Requirements:			

Perhaps my blasé attitude about reports on the web is because I've been using a perfectly acceptable – and not especially difficult to use – reporting mechanism and have been recommending it since I first wrote about it two years ago (as Figure 1 obviously demonstrates). I am, of course, referring to the lowly page, a.k.a. window procedure.

Marshall Brodine To The Rescue

One of my favorite quotes is from an old TV commercial by Marshall Brodine. He used to sell magic tricks (card tricks, actually) on TV and his tagline was "It's easy ... once you know the secret."

So, what's the secret to reports on the web?

Knowing the secret begins in understanding the medium, in realizing that the Internet can be used as your network but that the Web cannot. That is, it *is* possible to configure applications to log on to databases across the Net if you know the IP address. In this case, the Internet simply acts as your WAN (in fact, "WAN-dom" this is one of the Internet's original purposes and is how Citrix and ClarioNet work also). But as soon as you introduce a browser, the equation changes and changes in a very basic way. In a browser, everything must either be HTML or be HTML-based. (Technically, everything in a browser must be HTTP compliant, as that is how information is moved through the browser. In the end, it amounts to about the same thing.)

A baseline conceptual mistake many Clarion users continue to make is in thinking that Internet Connect and Web Builder simply project Clarion apps onto the Web (ClarioNet, though, comes awfully close). That is, Internet Connect and Web Builder are believed to somehow (magically?) turn the browser into a frame containing a standard Clarion app. In fact, Internet Connect and Web Builder "convert" standard output (video memory) to HTML (ASCII files that a browser "understands"). This extends a web server so that an image of a Clarion app can be presented in a browser.

There's a fundamental difference between these two ways of thinking about an application. In the first, it is easy to continue believing that the ACCEPT loop is still active; in the latter, it is much harder to do so.

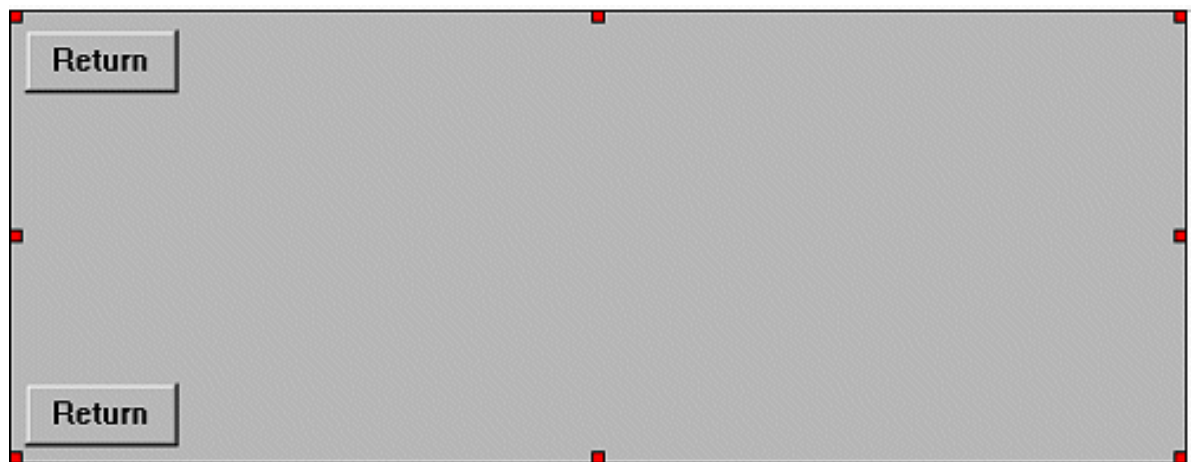
The bottom line is that when designing for the Web, it is important to constantly remind oneself that the finished product is going to be and, indeed, must be HTML. And, further, that HTML is presented in a "page." Technically, a page is just an ASCII file, a fancy ASCII file, an ASCII file with HTML headers but an ASCII file nonetheless.

NOTE: There are alternatives to HTML reports, such as PDFs. Although PDFs are perfectly legitimate, as hard as it may be to believe, I have seen machines without Acrobat Reader. The point here is that browsers (and therefore Clarion web apps) do not inherently print the way Windows desktop applications print.

Pragmatically, a page corresponds to a window. That is, in HTML, there are no browses, forms, processes or reports. In fact, there are no procedures, there are just pages.

A picture, they say, is worth a thousand words. So, look at Figure 4.

Figure 4. Window template procedure.



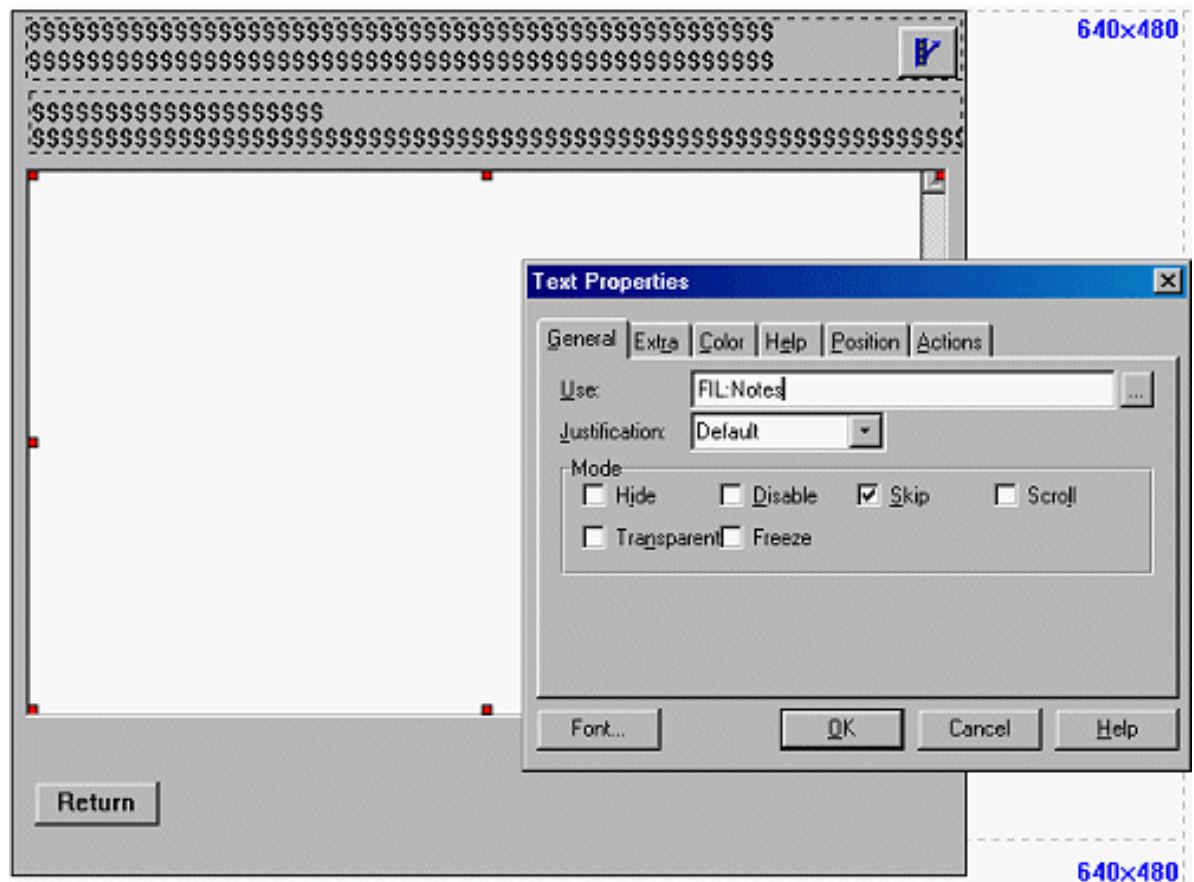
Now look back at Figure 2. The window in Figure 4 (design time) *is* the report in Figure 2 at runtime.

And The Secret Is ...

Ignoring the obvious trick of using a memo or text field, as in Figure 5, below, the secret is that you can produce your own HTML, essentially dictating the contents the ASCII file instead of simply accepting what Clarion produces (that's the sort of thing programmers do, right?).

Deep, dark secret, eh?

Figure 5. Using a Memo.



Except for the fact that the window skeleton provides the basic HTML wrapper (header, footer and body tags), reports are little different than exporting a file or subset of a file to ASCII (and this is something we all know how to do). This exception is a very important one, however, convenience-wise. That the skeleton, `Window.HTM`, provides all the required HTML *and* ensures that the broker handles the page creation, delivery and disposal correctly can save the need to use external tools, like CGI, and a bunch and a half of aggravation.

Using the built-in capabilities, what remains is formatting the information and putting it into the page in the appropriate place. And this is the easy part: `Target.WriteLine` (my second favorite web method). `Target.WriteLine` is a method used in both the Internet Connect and Web Builder TextOutput Class to write a line of text (HTML).

There are two ways I have implemented reports using `Target.WriteLine`.

(1) I have written each line as I formatted it (Figure 6).

Figure 6. `Target.WriteLine` one line at a time.

```

Target.WriteLine('<<center><<table border="0"><<tr><<td align="left"><<strong>')  !write HTML
Target.WriteLine('<<pre>** ' & LIS:Number & ' *** ' & LIS:TCODE & ' ***** <<font face="Verdana,
Target.WriteLine('<<br>')
Target.WriteLine('      ' & LOC:CompName & ' Contact: ' & LOC:ContactPers & '')
Target.WriteLine('      ' & LOC:Division & ' {10}' & LOC:Cont_Title & '')
Target.WriteLine('      ' & LOC:Address & '      ' & LOC:Word1 & LOC:How & '')
Target.WriteLine('      ' & LOC:CityStateZip & ' ' & LOC:Word2 & ' ' & LOC:When & '')
Target.WriteLine('      ' & LOC:Phone & ' ' & LOC:Extension & ' ' & LOC:Phone2 & ' ' & LOC:Extension
Target.WriteLine('      ' & LOC:Email & '')
Target.WriteLine('<<br><<font size="3">' & Clip(LOC:JobWords) & '<</font>')
Target.WriteLine('      Job Title: ' & LIS>Title & '')
Target.WriteLine('      Job Type: ' & LIS>Type & '      Location: ' & LIS:Location & '')
Target.WriteLine('<<br>')
Target.WriteLine('      Salary: ' & LIS:Salary & '')
Target.WriteLine('      Schedule: ' & LIS:Schedule & '')
Target.WriteLine('<<br>')

```

The final output for this is shown in Figure 2.

(2) I have also concatenated everything I wanted to display into a single string and

```
Target.WriteLine(DisplayString)
```

the whole thing, as in Figure 7.

Figure 7. Target.WriteLine a concatenated string.

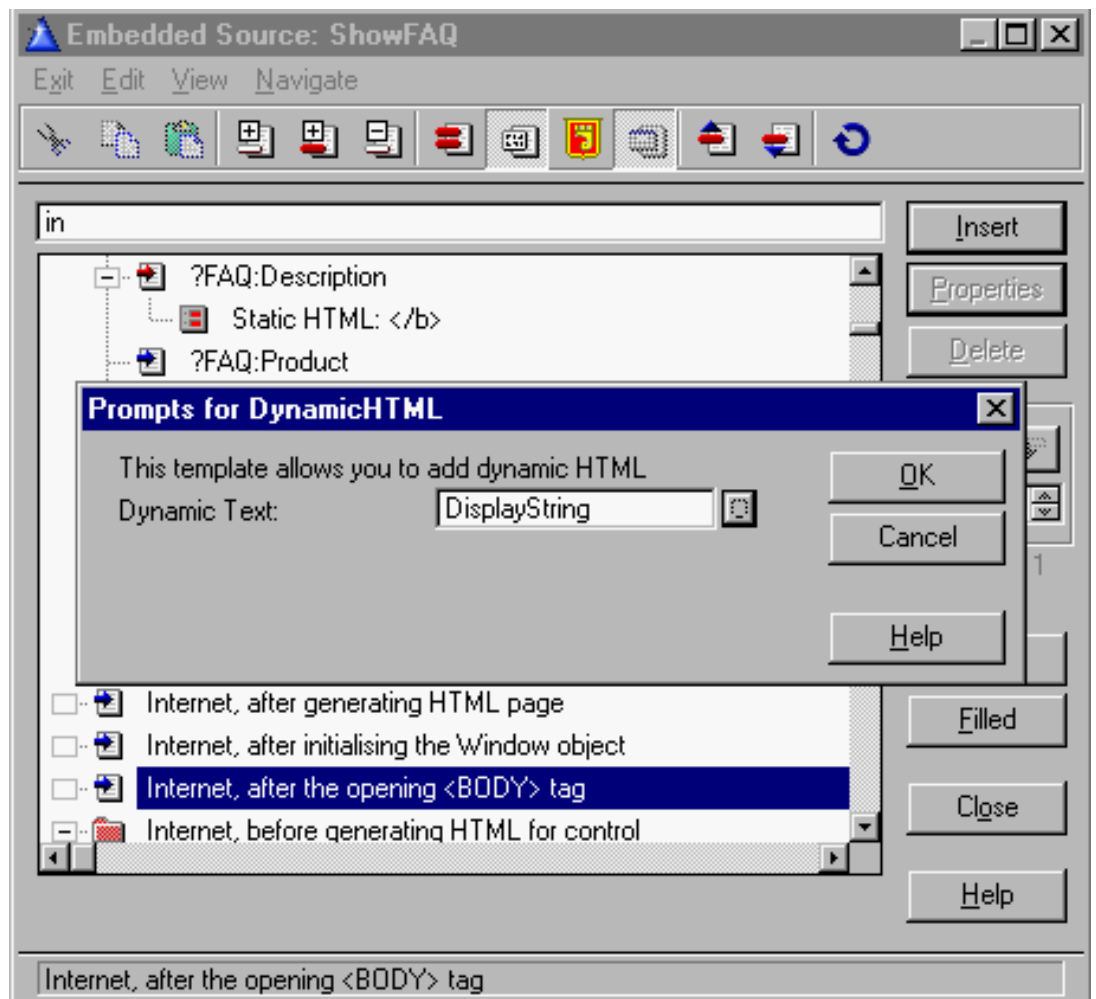
```

      Else
        DisplayString = DisplayString & Tempstring[i#]
        IsSpace = False
      End
    End
  End
End
Else
  DisplayString = Tempstring
End
Target.WriteLine('<<pre><<p>&nbsp;<</p>' & Clip(DisplayString) & '<</pre>')

```

With a single string like this, it is also possible to use the Dynamic HTML code template:

Figure 8. Using a Code Template to write HTML.



In the first case, I used an old Clarion for DOS report as the model for what I wanted the final output to look like. I requisitioned (politically correct expression for "write once, use many times," i.e., steal) most of the existing DOS code and simply counted spaces to lay out the controls in the same relative locations. I then opened the resulting dummy HTML in a visual HTML editor and beautified it a little.

In the second case, I have to read each and every character in the input string. Some of the input contains strings that are sample HTML. If I simply output the string, the browser would interpret it as HTML (well, it *is* HTML) and act on it accordingly. The HTML standard provides for substitution characters for <, >, &, *et.al.*, for example. So, I read the data, substitute as required and output a new string. The new string is what I `Target.WriteLine`.

But...

Ok, if no special formatting is needed, a memo or text field will do. And, if handling special characters is necessary, there is no alternative to string slicing. But, the example in Figure 2 looks like an awful lot of work (if you consider an hour or two "a lot," it is).

It looks almost as if I used one of those old 80 column coding pads for the layout. (I did.)

And, if a proportional font were used, all that fine formatting would be shot to hell in an instant. (It would, even using the `<pre>` -- preformatted text -- tag.)

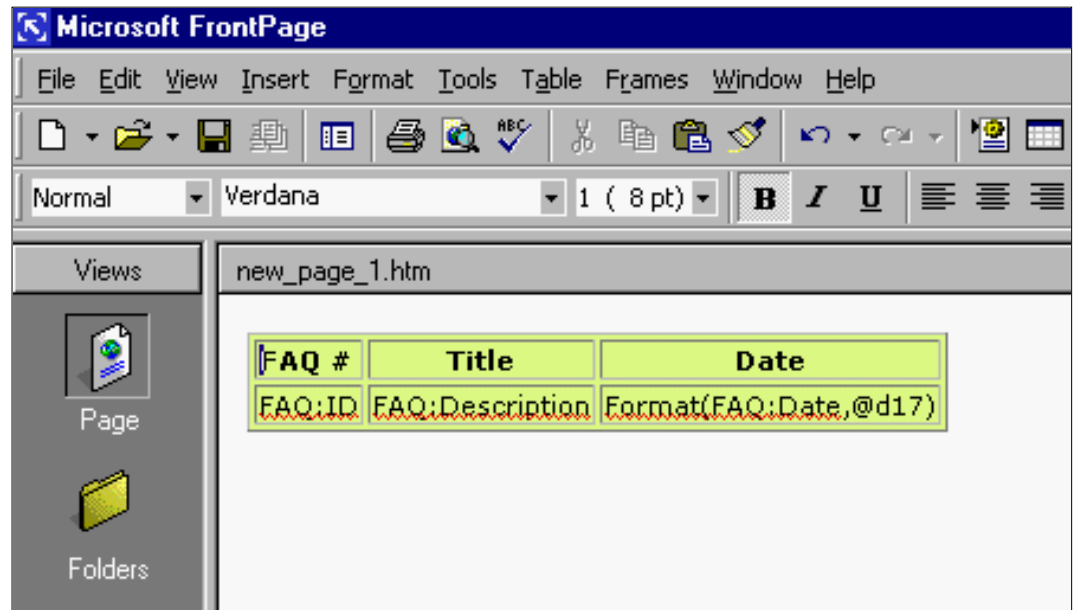
But, for the purpose, it is just what I want.

The alternative is to use my equivalent to the lost wax casting method, the free HTML

method. I fire up my favorite visual HTML editor and create an image of the output that looks like what I want.

However, instead of displaying data, I use real field labels.

Figure 9. "Report" laid out in FrontPage.



Viewing the HTML source shows that the code looks something like this:

Figure 10. HTML produced.

```

<table border="1" bgcolor="#E4FE8B"><tr>
  <td><font size="1" face="Verdana"><strong>FAQ #</strong></font></td>
  <td align="center"><font size="1" face="Verdana"><strong>Title</strong></font></td>
  <td align="center"><font size="1" face="Verdana"><strong>Date</strong></font></td></tr>
<tr><td><font size="1" face="Verdana"> FAQ:ID </font></td>
  <td><font size="1" face="Verdana"> FAQ:Description </font></td>
  <td><font size="1" face="Verdana"> Format(FAQ:Date,@d17) </font>
</td></tr>
</table>

```

Well, everything except my labels is just string data. And, I know how to deal with string data: It is surrounded by apostrophes and left angles are doubled up ('<' to '<<'). After that, I just concatenate the strings with my labels and Target. WriteIn the line, just as shown in Figure 6.

If, as in this case, the data is repetitive, I put it in a LOOP.

Summary

I don't often revisit a topic (actually, I usually forget I've dealt with it) but with the high level of interest in Clarion's new internet products, it seemed advisable to take another look at reports.

The free HTML method is an easy way to do reports even though they have not been officially implemented yet. In fact, entire applications can be built this way (the CWICWEB download app is done this way, allowing totally different look and feel on

the Web).

On most pages, I leave the "Cancel" or "Close" button and, on forms, I leave the "Ok" button. In essence, you can use your HTML editor for layout and Clarion for file access. It's not totally RAD but it *is*, in the immortal words of my esteemed editor, RFAD (Reasonably Fast Application Development). Better still, it works.

[Steve Parker](#) started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitor's right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source
Project](#)
[Issues in
PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)**TopSpeed****Clarion 5**
by TopSpeedFREE
Microsoft
Internet
Exploreretc2000
EVENT SPONSOR

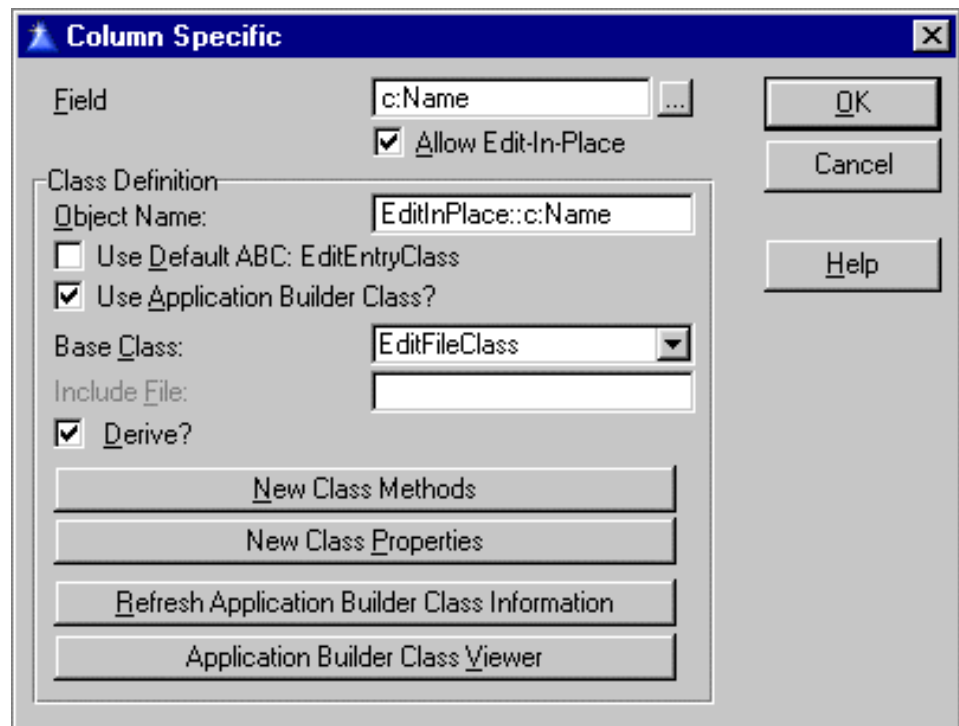
The Clarion Advisor: EIP File Lookups

by Russ Eggen

If you use the EditFileClass to lookup a file, the entire path is returned. This is not a problem if the entire file path is what you want, but sometimes you only need the file name itself.

To get around this, I made a new method in this class. When entering the Column Specific EIP class settings, click on the Derive button, which will enable the New Class Methods and New Class Properties buttons, as shown in Figure 1.

Figure 1. Deriving a class method.



Click on New Class Methods, and add a method called FileOnly with a prototype of
(STRING xPathName), STRING.

[Web Builder Reporting](#)
(Jul 5,2000)[The Clarion Advisor: EIP
File Lookups](#)
(Jul 5,2000)[Please Mr. Postman:
Calling External Functions
By Address](#)
(Jul 5,2000)[July 2000 News](#)
(Jul 5,2000)

Add this source to it:

```
IF INSTRING('\',xPathName,1,1)
  Len = LEN(CLIP(xPathName))
  LOOP Ndx = Len TO 1 BY -1
    IF xPathName[Ndx] = '\' THEN BREAK.
  END
  RETURN xPathName[Ndx + 1 : Len]
ELSE
  RETURN xPathName
END
```

In the method's data embed:

```
Len USHORT,AUTO
Ndx USHORT,AUTO
```

Close all dialogs and go to the embed tree.

In the EditFileClass/CreateControl method, you can place this code (salt to taste):

```
SELF.Title = 'Choose an image'
SELF.FileMas
k = FILE:LongName SELF.FilePattern = 'GIF images|*.GIF'
```

In the object's TakeEvent embed you can add this code to show only the file name:

```
SELF.UseVar = SELF.FileOnly(SELF.UseVar)
DISPLAY
```

It may not be the prettiest or the most elegant, but it does work very nicely. If you have a faster/easier way, send it to editor@clarionmag.com. That's a challenge!

By the way, if anyone is wondering where I figured out to use SELF.UseVar, it is listed in the ABC viewer. This code won't work with a field equate or field name.

[Russ Eggen](#) has been using Clarion since 1986. Before joining Topspeed as a consultant in 1996, he was an independent contractor. Currently, Russ is an instructor at SoftVelocity and is writing the curriculum for the classes. His main goal in life is to get a Clarion program to star in a Tom Clancy movie where the program helps the hero save the world.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To](#)
[Other Sites](#)[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)

Please Mr. Postman: Calling External Functions By Address

by **Steve Bottomley**

This topic falls squarely in the "more than one way to skin a cat" category. I'm presenting the Nothing But Clarion method of calling external functions by address.

First, a bit of history.

Even before Clarion for Windows Version 1.0 went gold, developers were asking how to access functionality built into the Windows API (Application Programming Interface). First, there was no RUN statement in that prehistoric era so they wanted to call the WinExec API function (we were 16 bit only back then) to start another program. Then they discovered that calling WinExec wasn't enough because their program would keep right on running instead of waiting for the new program to finish. So they wanted to call GetModuleUsage in a timer loop in order to obtain the proper behavior. A few polite messages to the Development Centre, a bit of digging around by various developers and lo and behold, we found these things were all possible.

But wait. If it was possible to access Windows functions, why not access functions from other programs? More polite messages to the Development Centre, more digging around and the secrets to using non-Clarion DLLs as well as creating DLLs in Clarion for use by other applications were brought to light and, over time, made even easier by TopSpeed. Libmaker was born, WinEqu.clw began shipping with Clarion, and a new artform was created: converting header files from C for use with Clarion.

C header files (files with the extension .h) are like the space before the CODE statement in a .CLW file. They contain data definitions and function prototypes. It was for the most part (and macros aside), reasonably simple to convert these headers. A #define would convert to a Clarion EQUATE, a struct would convert to a GROUP and functions could be prototyped using a few simple rules like adding the RAW, PASCAL or C attributes where required. Once you'd done the conversion and created a LIB file to add to your project (using LibMaker), you could quite happily call those external functions the same way you would call any Clarion procedure or function: that is, by the label given to it:

```
ReturnValue = SomeExternalFunction(SomeParameter)
```

[Web Builder Reporting](#)
(Jul 5,2000)[The Clarion Advisor: EIP File Lookups](#)
(Jul 5,2000)[Please Mr. Postman: Calling External Functions By Address](#)
(Jul 5,2000)



In the words of Ross Santos, "Kewl!"

The Problem

Like life, programming in any language, including Clarion, is not always as straightforward as it seems. Let's say you've decided to implement functionality from another programmer and, like a good little bithead, that person used C. Not a problem, all you have to do is convert a few prototypes to Clarion notation, include the .LIB file and away you go.

You're well into the conversion process and all of a sudden you're faced with something like.

```
typedef FUNCPTR(LRESULT, FCAPIV, PMSGPROC)(UINT unMsg, ...);
or
typedef BOOL (WINAPI *dsAddIt) (
    DWORD dwHandle, BS_CFG *Cfg );
```

You check the documentation and, if it is of better-than-usual quality, it mentions something like "as part of the initialization process procedurex returns a pointer to PMSGPROC.... A pointer? What good is that! A pointer is an address, and Clarion doesn't call procedures by address.

You look at the DLL using LibMaker and this particular function isn't listed there, so it hasn't been exported. This means that even though you know the prototype, Clarion can't use the .LIB file to resolve this particular function when you need to call it. You can't use CALL because this procedure takes a parameter and returns a value. You now have a couple of options: spend some money; write some C; give up; or you can call the Postman.

The Postman

What is a postman? In the good old days of snail mail, you would give the postman a package bearing a destination address. The postman would deliver the package and, if required, would return with a reply.

In this context the postman is a procedure that will be given the address of the external function, any parameters it requires, and it will return with whatever the function passes back.

Sounds simple enough, but what's really going on?

Put simplistically Clarion is already passing procedures by address but the language (read compiler) hasn't been told to recognize a * in front of a procedure label in a prototype. The following would get rid of the need for the postman procedure entirely.

```
MODULE('external library')
    *ExternalProc(LONG), LONG, PASCAL
END
```

And the following would do away with the need to play hide and seek with the compiler.

```

MODULE( 'external library' )
    ExternalProc( LONG ) , LONG , PASCAL , TYPE
END
MODULE( 'some module' )
    Postman( *ExternalProc ) , LONG
END

```

Unfortunately neither of these options is available. Even though Clarion passes procedures by address, the compiler/linker needs an exported name for the procedure. So you have to trick the compiler.

The Trick

The secret to calling procedures by address is to use MAP statements at the module level (which is something ABC already does). This allows you to intercept the compile process before the MAP statement is read and redefine the data types used by any of the prototype parameters within the map.

Without this you would end up with "prototype mismatch" errors and your application wouldn't be created.

Now the how.

Below are the C prototypes for two functions in an external DLL, one that needs to be called by address (RECIPIENT), and one that returns the pointer to the RECIPIENT function (GetAddress). RECIPIENT has one parameter that is a LONG and it returns a LONG.

```

typedef LONG (WINAPI *Recipient) (LONG lParam);
typedef LPRECIPIENT (WINAPI GetAddress) (VOID);

```

First, you'll need a postman procedure. This is the only procedure that needs to know the exact prototype of the RECIPIENT function and its only purpose is to deliver the parameters and return any response. It would normally be a procedure created using the Source template type and in this case, its prototype is:

```

Postman(Destination ParcelDrop, LONG Contents1) ←
    , LONG , NAME( 'Postman' )

```

The postman procedure's prototype contains the label given to the external function's prototype as the first parameter and any other parameters that the function can receive. The prototype for the postman procedure also needs to have the NAME attribute in order to resolve the two different local prototypes that will be used. Without the NAME attribute, Clarion's name mangling will attribute two different "names" internally.

The postman procedure needs two additional pieces of code. The first is a module MAP statement that contains the prototype of the external function with the TYPE attribute added so the linker doesn't go looking for it at compile time. The label of this function can be anything although whatever label is used, must also be reflected in the first parameter of the postman procedure's prototype; I've used "Destination" in this case.

Note: In your application, the MAP entry would be added to the "Module Data Section" embed of the MODULE containing the postman procedure (select the Module tab on the procedure tree, then the .CLW that contains the postman procedure and then press the

"Embeds" button).

```
MAP
Destination (LONG Param1), LONG, PASCAL, TYPE END
```

The second piece of code is the actual source to call the external function and pass back anything returned.

```
RETURN(ParcelDrop(Contents1))
```

Remember that `ParcelDrop` is the name for the `Destination` function. As you can see, this looks suspiciously like calling any other external function for which you had a `.LIB` file.

Next, you need to tell the calling procedure how to call the Postman using the address returned by the `GetAddress` function and without getting any complaints from the compiler. This is where you intercept and redefine the datatype. If you're not using `ABC`, this requires the use of another module `MAP` statement in the calling procedure or, if you are using `ABC`, it just requires an `EQUATE` (the `ABC` templates automatically include prototypes for any child procedures defined in the application tree). Whichever one is used, it is placed in the Start of module embed of the calling procedure's `MODULE` (see above).

Without `ABC`:

```
MODULE('CALLA003.CLW')
  Postman FUNCTION(LONG ParcelDrop, LONG Contents1), ←
    LONG, NAME('Postman')
END
```

With `ABC`

```
Destination equate(LONG)
```

Note how in this case, instead of the label of the external function as the first parameter of the postman procedure, we are using a `LONG` (with `ABC`, the Generator still `INCLUDES` the postman procedure's prototype that uses the external function label, but we have now `EQUATED` that label as a `LONG`).

The calling procedure is now ready to get the address of the external function and call the postman to make a delivery and return with any reply:

```
DestinationAddress  LONG
Reply               LONG
  CODE
DestinationAddress = GetAddress() Reply =
Postman(RecipientAddress, 10)
```

To see this in action, download the [example code](#). You'll need to compile a small dll as well as the test app.

[Steve Bottomley](#) is a long time user of Clarion, a member of Team Topspeed, and works for the Australian Government.



Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)[Links To Other Sites](#)[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)**TopSpeed****Clarion 5**
by TopSpeed

View Wizard 1.0 From Nice Touch Solutions

Reviewed by Tom Hebenstreit, Reviews Editor

The browse is arguably the centerpiece of most data-oriented Clarion programs, and in its current ABC implementation, it is quite powerful. From the users standpoint though, it lacks two key features: customization and persistence.

Basically, when it comes to the fields and formatting of a browse list box, users only get what you thought to give them. Field X may be incredibly important to them, but if you didn't think it was important enough to put on the list, they are out of luck. If it just *has* to be added, you have to recode, recompile and redistribute the application. If you write commercial software, often the user simply has to live with the omission – or start looking for software that works closer to the way they want it to. Either way, the results are not what you want, i.e. happy users.

When it comes to persistence, well... there isn't any. If the user carefully adjusts column sizes to fit everything into the list, all of these tweaks disappear as soon as they leave the browse procedure. The next time they open the browse, it is right back to the defaults that you (the programmer) designed it with.

Yes, I know that I could write code to do all of this. Being a Certified Lazy Programmer (CLP), though, I'd much rather throw a little money at someone else and let them take care of it for me.

The benefits are threefold: my users get exactly what they want without further effort on my part; valuable project time and money isn't being wasted on re-inventing wheels; and I get to concentrate on the parts of the application that are actually unique and challenging. I guess you could call it a win-win-win situation.

All of this is a rather roundabout introduction to View Wizard from Nice Touch Solutions (whom I will hereafter refer to as NTS to save my poor, tired fingers). As you might guess, View Wizard was designed expressly to address the browse shortcomings that I have been talking about. Let's see how it does...

Major Features

[Product Review: View Wizard 1.0 From Nice Touch Solutions](#)
(Jul 11,2000)

[Sidebar Menus](#)
(Jul 11,2000)

[The Clarion Advisor: Memory Leaks And Virtual Methods](#)
(Jul 11,2000)

[July 2000 News](#)
(Jul 11,2000)



Rather than try to paraphrase a whole lot of items, here is a quote from the View Wizard help file:

View Wizard facilitates access to on-demand custom list box formatting and browse sorting from within any application. This process is easily integrated into existing or new applications by simply dropping the View Wizard browse extension template onto any Clarion Browse.

- *View Wizard supports development under Clarion 5 using either the ABC or Clarion (legacy) template chains.*
- *Provide end-users with the ability to modify (and save) a browse's list box layout and/or sort order.*
- *Saved views may be recalled via drop list controls, browse selection dialogs or programmatically.*
- *Optionally enable the end-user with the ability to hide/show columns.*
- *Optionally enable drag and drop column arrangement.*
- *Optionally enable user-definable dynamic sort orders....
Sort orders may be default, selected from existing keys, or completely ad-hoc (using fields of the end-user's choice). The dynamic sort order can be saved with the column arrangement to provide a set of views completely tailored by the end-user.*
- *Optionally enable column header sorting (ascending/descending).*
- *Optionally enable "smart" column width adjustments.*
- *View Wizard shares a very common interface with the NTS Wizard family of products.*

As you can see, View Wizard provides the user with fairly comprehensive control over the contents and order of their browses. What sets it apart from other similar products is the wizard-oriented nature of the product. Rather than presenting a complex dialog to the user, it provides a step-by-step process of selecting fields and specifying orders.

As you may have noticed from all of the optionally enabled features on the list, as a developer you have very fine control over just which features, fields and sort options are available to the end-user. This type of flexibility lets you tailor the use of the product to match your specific needs on a browse-by-browse basis (View Wizard can accommodate multiple browse boxes in a single procedure).

Additionally, building your browses can require a lot less effort on your part, as now you can leave many of the final details in the hands of the users (where it actually belongs). All you have to do is give them the framework. You can also use View Wizard yourself to build as many custom views as you want, each one with unique combinations of columns and sort orders. Then just ship them as default views with your product.

Installation

After purchasing View Wizard, you are given a registration number that allows you to download the product from the NTS web site.

Once you have View Wizard, installing the product is about as easy as it can get. The setup program asked for my registration information and then basically took care of everything else automatically. It found the correct Clarion folder for the version I was installing, registered the templates for me and then finished by displaying the View

Wizard help file so that I could get oriented with the product.

Upon reinstalling the product it even pre-filled my registration information, reducing the process to a grand total of six mouse clicks. Very nice, indeed.

As for improvements, my main suggestion would be to allow the user to choose the program group where the View Wizard shortcuts will be placed. To avoid having a bazillion items on my Windows Start->Programs menu, I tend to want to group things into other sub-menus. For example, I have all of my Clarion 5 third party tools grouped together in a 'Clarion 5 Accessories' menu. The View Wizard installer placed its group at the higher level without offering me any choice.

Why not just move the menu items myself? That's what I ended up doing, but you need to be aware that moving a menu item yourself means that it will no longer be removed by an uninstall program. And speaking of uninstallers, View Wizard does not have one.

My other suggestions are related to the View Wizard demo application. First, it is not mentioned at all in either the setup program or the help file. Secondly, it is installed under the base Clarion5 folder, rather than in the common Examples or 3rdParty\Examples folders. I'd like to see at least some notice that it is there, and have it placed in what seems to me a more logical location.

All in all, though, the installation process is painless, complete and very straightforward.

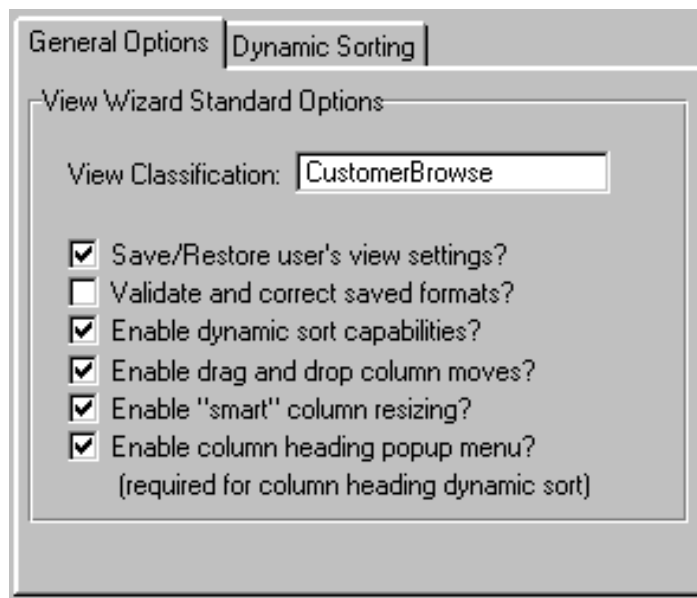
Implementation

Templates, that's all it takes. Add a global extension, drop a control template or two onto your browses, check some boxes to indicate how many features should be available, and you are pretty much done.

At the most basic level, you can set View Wizard up so that the users never see the wizard (I know, it sounds a bit strange, but bear with me). In this mode you can, for example, allow the user to customize the list by dragging and dropping columns, hiding and restoring columns, doing a single level sort on any column and more -- plus their changes are persistent without the user having to do anything else.

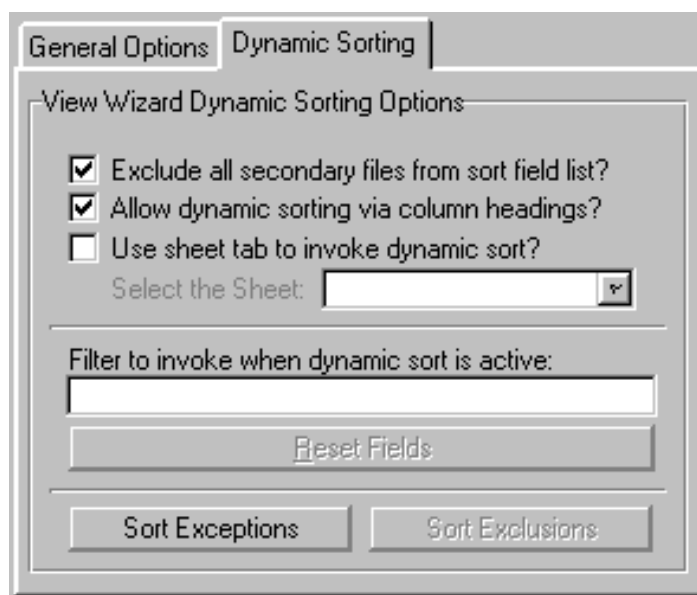
All of this is done on the browse itself via the right-click pop-up menu, or by simple drag and drop. The user is limited to one browse style and can only use the fields that you as programmer have placed on the browse. Figures 1 and 2 show the template options you would use to do this.

Figure 1. View Wizard General Options.



In case you are curious, the View Classification field is merely a unique name that View Wizard uses to keep track of which saved browse setups belong to which browse.

Figure 2. View Wizard Custom Sort Options.



Note that you can decide which fields your users can use as sort fields.

A logical next step up is to allow the users to create, name, save and restore multiple browse views, and it is here that you would normally provide access to the actual View Wizard. Your biggest choice at this point is deciding how the user will access the wizard.

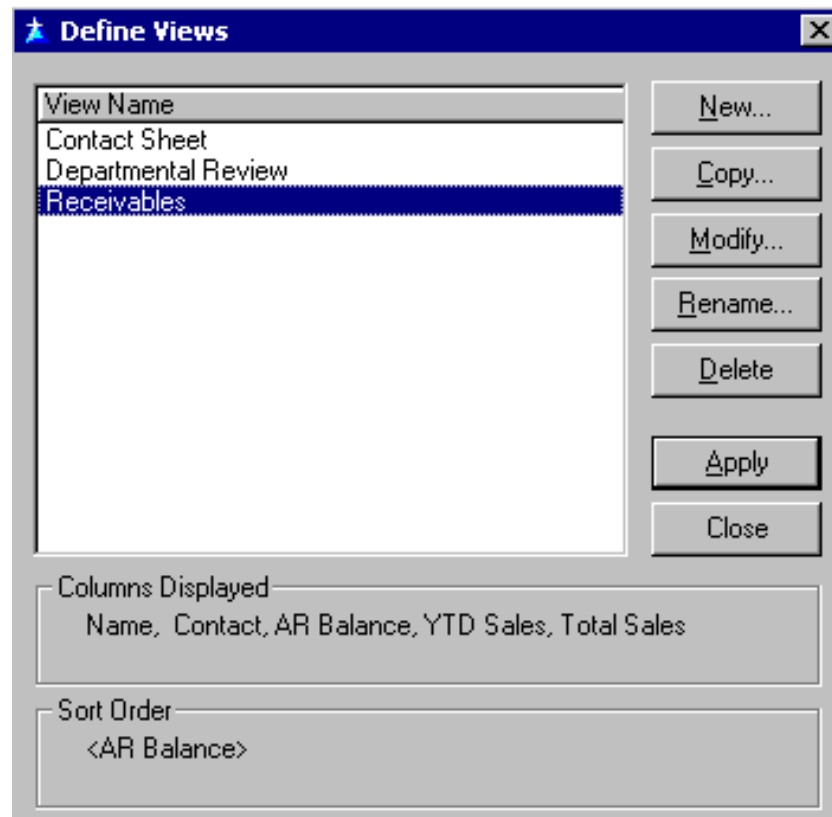
There are three basic methods of implementing user access: buttons, a drop-list, and the list box right-click context menu.

As far as the programmer is concerned, the button and context-menu methods are the same: you place a View Wizard control template on the window, and it populates the three View Wizard buttons (Modify View, Reset View, Define Views). If you also want to enable pop-up context menu support, you check a box on the template prompts – now the user can use either method. If you only want them to access View Wizard from the

pop-up menu, just hide the buttons.

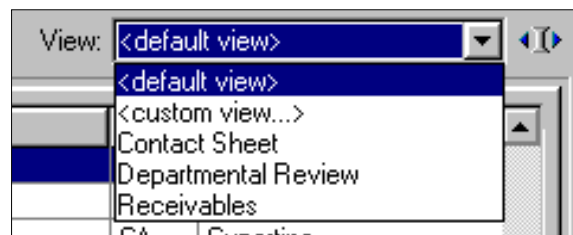
The alternate method is to use a drop down list box on the browse that lists all of the available view orders. Two additional items are always on the list: '<default view>', which resets all view and sort parameters back to your original defaults, and '<custom view...>'. The latter is the option that provides access to the actual View Wizard. Once again, if you also want to allow pop-up menu support, just add the button template and hide the buttons.

Figure 3. The Define Views window.



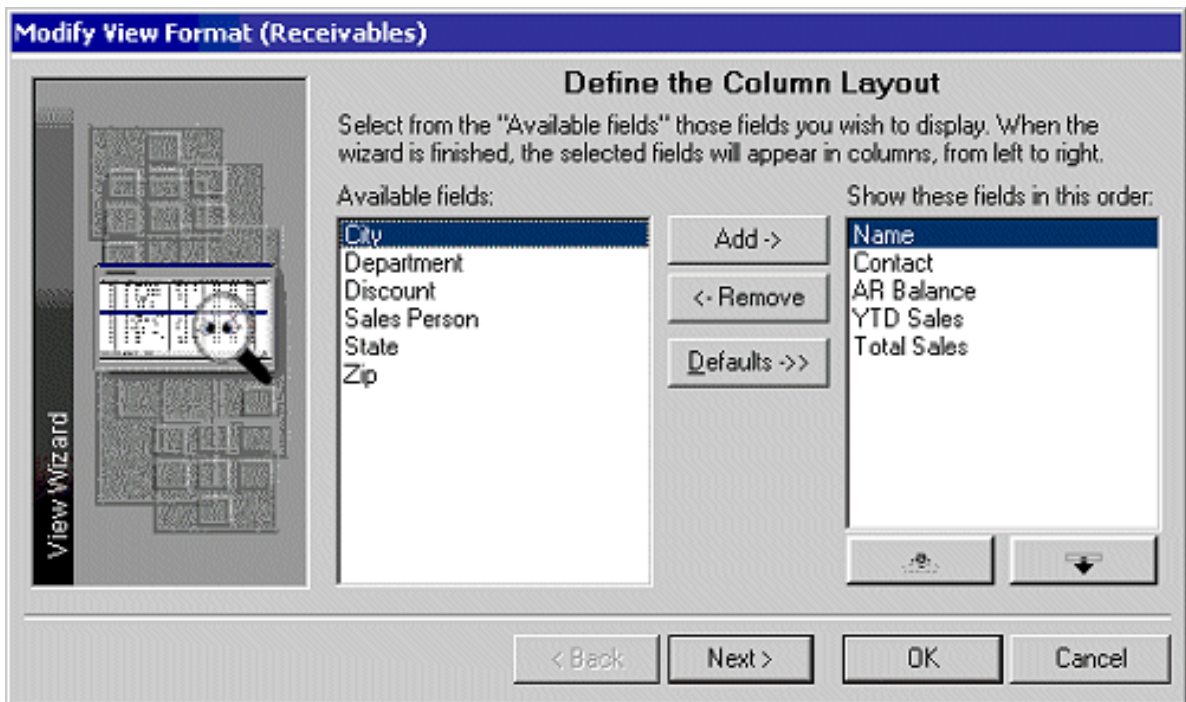
If you aren't using the drop list, Figure 3 shows the primary interface your users will use to create, maintain and select their custom views. Note how it shows the displayed columns and sort order for the selected view.

Figure 4. Selecting a View from the drop list.



Note the button to the right of the drop list. It allows the user to immediately invoke the View Wizard without having to drop down the list.

Figure 5. A panel from the View Wizard itself (selecting fields).



As you can see, you have a tremendous amount of flexibility in determining both how View Wizard behaves and when it appears to your users.

NOTE: Because there is no easy way for the user to see exactly what the currently selected view and sort is in some modes, NTS also provides two code templates that will place the current view type and sort order into strings that you can then display on the browse window. An additional template provides for the use of a custom locator when a user-defined sort is active.

Performance

View Wizard is very easy to use, and things just seem to work the way you expect.

For example, if you are in a named/saved view and change something (move or hide columns, for example), the name of the view changes to '<custom view>' to reflect the fact that you have made changes. If you pop up the View Wizard after hiding a number of columns, the View Wizard reflects the changes you just made. If you want to unhide a column you've hidden, View Wizard automatically places all hidden columns on the 'Show Column' list in the pop-up menus.

View wizard also adds some nice Windows Explorer-type touches to your browses, such as the ability to automatically size a column to the width of the largest entry. Just double-click on the vertical line between column headings (the one that you grab to manually resize columns.)

As a final bonus, the simplicity of the one-thing-at-a-time wizard style presentation means less confusion on the part of the user, and that can translate to fewer technical support demands on your time.

Documentation

Documentation is provided in the form of a standard Windows Help file. It does a good job of explaining what View Wizard does, and includes a handy Quick Start section that

walks you through adding View Wizard to an application. A Frequently Asked Questions (FAQ) section and complete explanations for using each of the templates are also in the help file.

View Wizard also includes a runtime help file that is oriented towards the end-user, explaining the actual wizard screens.

One ABC demo application is included with View Wizard. After taking a look at it, it appears to be the source for the downloadable View Wizard demo program. It is useful on two fronts: first, to show how to fill in the template prompts to achieve the desired level of functionality in the wizard, and second, to illustrate how little source code is needed to actually implement View Wizard (in a nutshell, none).

About the only thing missing (as I mentioned above) is any mention of the demo application in the documentation. If I hadn't gone looking for it, I wouldn't have known it was there.

Technical Support

Free support is provided via email (up to reasonable limits, of course), and NTS also offers paid telephone support. Support from NTS has been uniformly excellent in all of my dealings with them, and in this case my various questions regarding View Wizard were always answered on the same day.

Summary


Probably the only question left is what View *doesn't* do. The only obvious missing feature I can see is that you can't just click on a column heading to sort by it. Even this is no big shakes, as you can do your sorting from a right-click pop-up menu, but it is the only place where View Wizard doesn't follow the behavior that a user might expect to see. NTS plans to add this feature for the next major release.





If you need even more control over the look, feel or behavior of View Wizard, NTS offers the complete source code for a very reasonable price (see below for pricing information). If you do SQL development and want to fully integrate the View Wizard saved views storage into your own database, buying the source code is pretty much a requirement.

Bottom line: View Wizard is easy to implement, easy to use, and can give your applications an instant boost in user friendliness.

I had no problems using it, and it required absolutely minimal effort on my part (no embed code!). View Wizard has a very polished look to it, and the fact that it integrates seamlessly with the other products in the NTS Wizard line (e.g., Query Wizard, Report Wizard, et al.) is icing on the cake.

Download the demo and try it out – I think you'll like it. I know your users will!

PRODUCT RATING	
Overall	
Ability to do the task	Excellent
Ease of use	Excellent
Ease of installation	Very Good
Documentation	Very Good
Technical support	Excellent
Black box DLLs/LIBs	Yes*

LEGEND	
First class all the way	
More than adequate	
Barely adequate	
Don't even think about it	

* Complete source code for the View Wizard can be purchased separately.

View Wizard is currently available for Clarion 5 only, using either the legacy or ABC template chain. NTS expects to release a 5.5 compatible version as soon as SoftVelocity releases the final version of 5.5. According to NTS, customers with the View Wizard sources have already recompiled it for 5.5 with no problems.

View Wizard lists for US\$229, and is available direct from Nice Touch Solutions. Source code for View Wizard is an additional US\$99, and yes, you must be a registered owner of View Wizard in order to purchase the sources. View Wizard is also available at a discounted price in various product bundles offered by NTS. Please visit the NTS web site for full pricing and ordering information, as well as further product information and downloadable demos:

<http://www.clariontools.com/clarion/default.htm>

A longtime Clarion user, [Tom Hebenstreit](#) is an admitted tool junkie who refuses to go straight and code without his arsenal of third party products. During those rare moments when he isn't either using or writing about Clarion, he indulges his twin passions for blues and beer by performing around Southern California in a variety of totally-obscure-but-famous-any-day-now rock and blues bands.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)**TopSpeed****Clarion 5**
by TopSpeed

Sidebar Menus

by Steve Parker

There have been a number of requests for a side-menu-bar-next-to-the-work-area style app frame. Internet Explorer sports such an interface when, for example, you click on either the Favorites or History icons. Netscape 6 defaults to this type of layout.

I never understood the appeal of this layout. It seemed to me no more useful than a top toolbar, though occasionally better looking (more room for fancy icons, you see), and I always found that it cut into my work space.

Last year, however, a customer took a standard browse, not a bad looking one (not a particularly good looking one either), and maximized it. He likes *everything* maximized. Visualize a 15 or 16 line browse box with the standard buttons (plus a few more) at 1024 x 768. Now picture it maximized. Not an especially appealing picture, this.

Now picture it on a 20 inch monitor.

Now I see a purpose for a vertical menu bar. By using up some of the screen real estate, a maximized browse is ... less unattractive. Compare Figures 1 and 2.

Figure 1. Standard browse maximized at 1024 x 768.

[Product Review: View Wizard 1.0 From Nice Touch Solutions](#)
(Jul 11,2000)

[Sidebar Menus](#)
(Jul 11,2000)

[The Clarion Advisor: Memory Leaks And Virtual Methods](#)
(Jul 11,2000)

[July 2000 News](#)
(Jul 11,2000)

Developer
PLUS

Your source for
development tools
and add-ons.

Your outlet for
application sales.

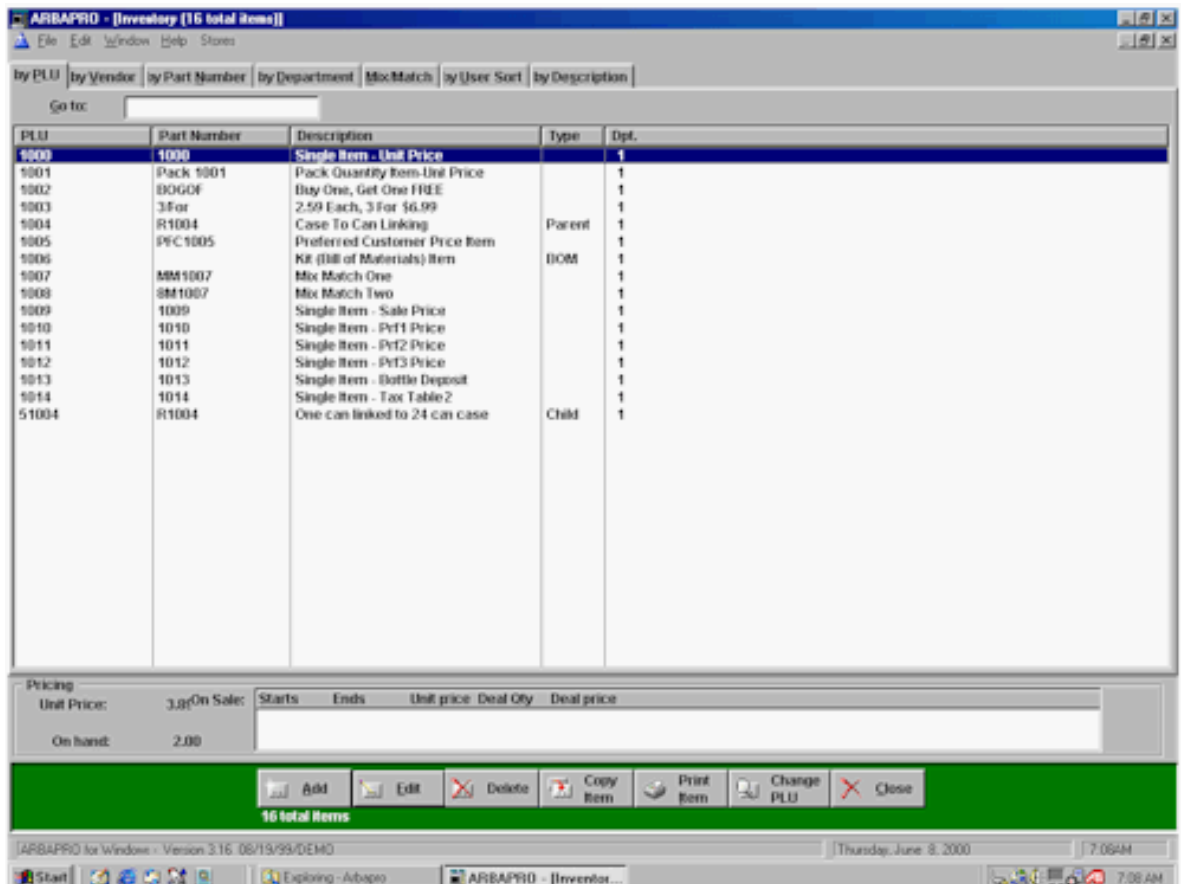
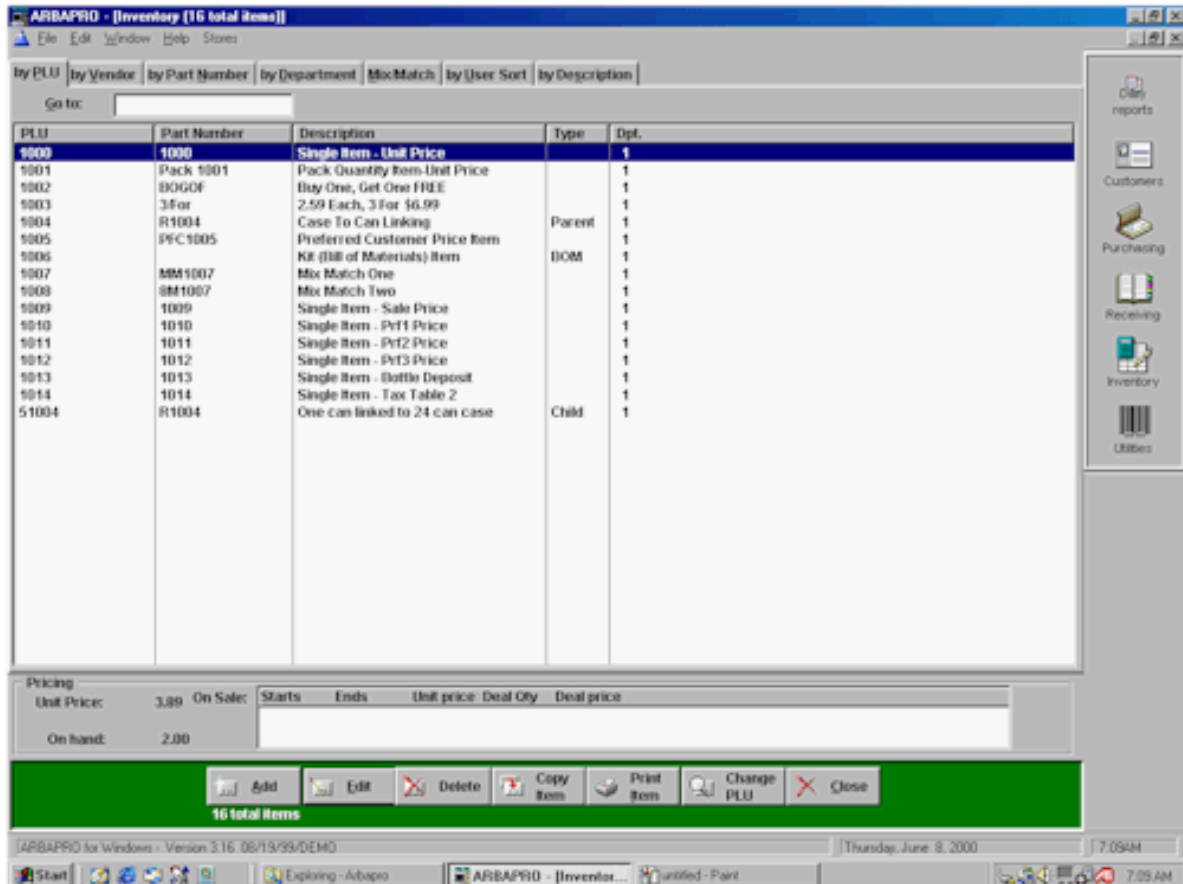


Figure 2. Browse with vertical menu.



Elimination of that huge white space at the right of the browse box is certainly an

improvement. The buttons, at the bottom, also don't look quite so lonely.

But, what about end users with lower resolution monitors? A recent survey indicates that 56% of business computers support only 800 x 600 and a still substantial percentage are only standard VGA. And what about those teeny-tiny, 9 inch POS (Point Of Sale, in case you misunderstood) monitors?

What if a window cannot be sized to fit at lower resolutions with a side menu open (as, indeed, happened to me)?

In these cases, I want to be able to close the vertical menu to recapture the horizontal space. I also need to be able to re-open the vertical menu when the larger window is closed. Well, actually, rather than automate this, I want to put it under user control. That way, the user can open and close the vertical menu. (Maybe the user never wants to see it; I can "Ini-file-ify it." Translation: "I don't have to be bothered.")

So, I see three tasks I have to accomplish:

- Create a vertical menu
- Create frame menu items to open and close it

and

- Turn the appropriate menu items on and off

Vertical Menus

I immediately rejected a toolbar control because a toolbar cannot be sized horizontally. Specifically, while a toolbar's height can be changed, its width cannot.

It turns out that a vertical menu is quite simple; a window will do.

I populate a series of buttons on the window and have them call procedures. Simple enough.

The "trick" is in using the window's toolbox attribute.

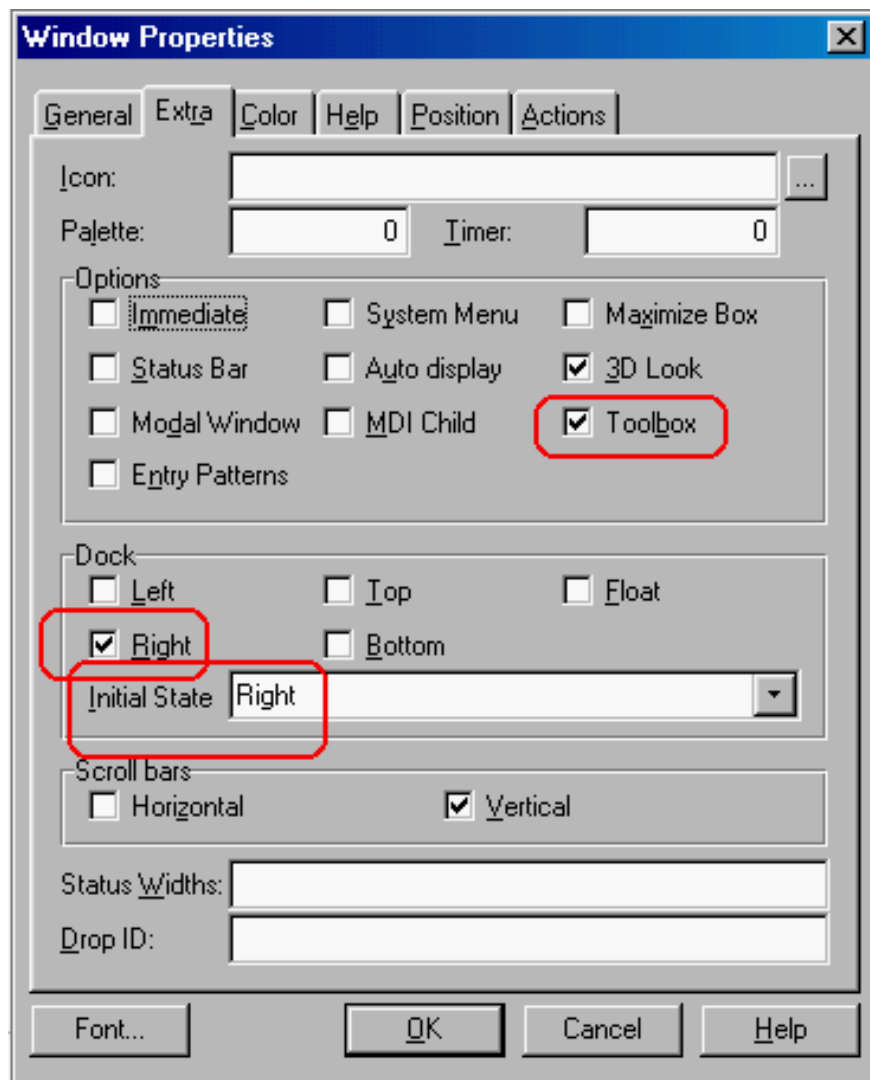
The Language Reference tells us:

The **TOOLBOX** attribute (**PROP:TOOLBOX**) specifies a **WINDOW** that is "always on top" and may be docked if the **DOCK** attribute is also present. Neither the **WINDOW** nor its controls retain input focus. This creates control behavior as if all the controls in the **WINDOW** had the **SKIP** attribute.

The next thing is to allow the window to dock and to do so only in one place (in the case shown in Figure 2, on the right). I want it to dock because a toolbox is always on top. Undocked, it would end up on top of my windows. Users would be forced to move it out of the way (and many don't even know what a mouse is, much less how to use one!). I want it in one place so it looks like a real menu (*real* menus don't move around) and users can accommodate to its location.

To ensure this happens immediately, whenever the toolbox window opens, I set that as the window's initial state:

Figure 3. Window properties to create a vertical menu.



A window with the toolbar attribute cannot be resized and, because of the options I have chosen it can only dock in one place and cannot be moved. That's all there is to it.

Further, because, as the Help says:

Normally, a WINDOW with the TOOLBOX attribute executes in its own execution thread to provide a set of tools to the window with input focus.

it is probably possible to populate a tree control in the toolbox window and have it serve as a collapsible menu. I have seen this but not tried to duplicate it. It is likely that you will have to hand code the required queue or, you could create a set of preloaded files that the user cannot modify. You would almost certainly have to hand code the action handling for the tree. But this is not rocket science, and has been covered in a previous Clarion Magazine [article](#). Result: an instant (or almost instant) Outlook-style menu.

Opening And Closing The Toolbox

To begin with, go to the online help and look for How Do I ... | Windows | How to Manage Threads. This section provides the basic information necessary to open and close the toolbox.

There are four things I need to know in order to open and close the toolbox.

1. I need to know when the user wants to close (disable) the toolbox.

2. I need to know when the user wants to open (enable) the toolbox. There will be menu items on the main frame to open/close the toolbox.
3. I need to know the thread the toolbox is on. I need this so I know which thread to post open or close events to. (To jump ahead a bit, it turns out that I really only need to know the thread for closing the window; stay tuned.)
4. Because I will be calling events affecting a menu on the app frame, I need to know the frame's thread (that is, I need to know where the menu items are).

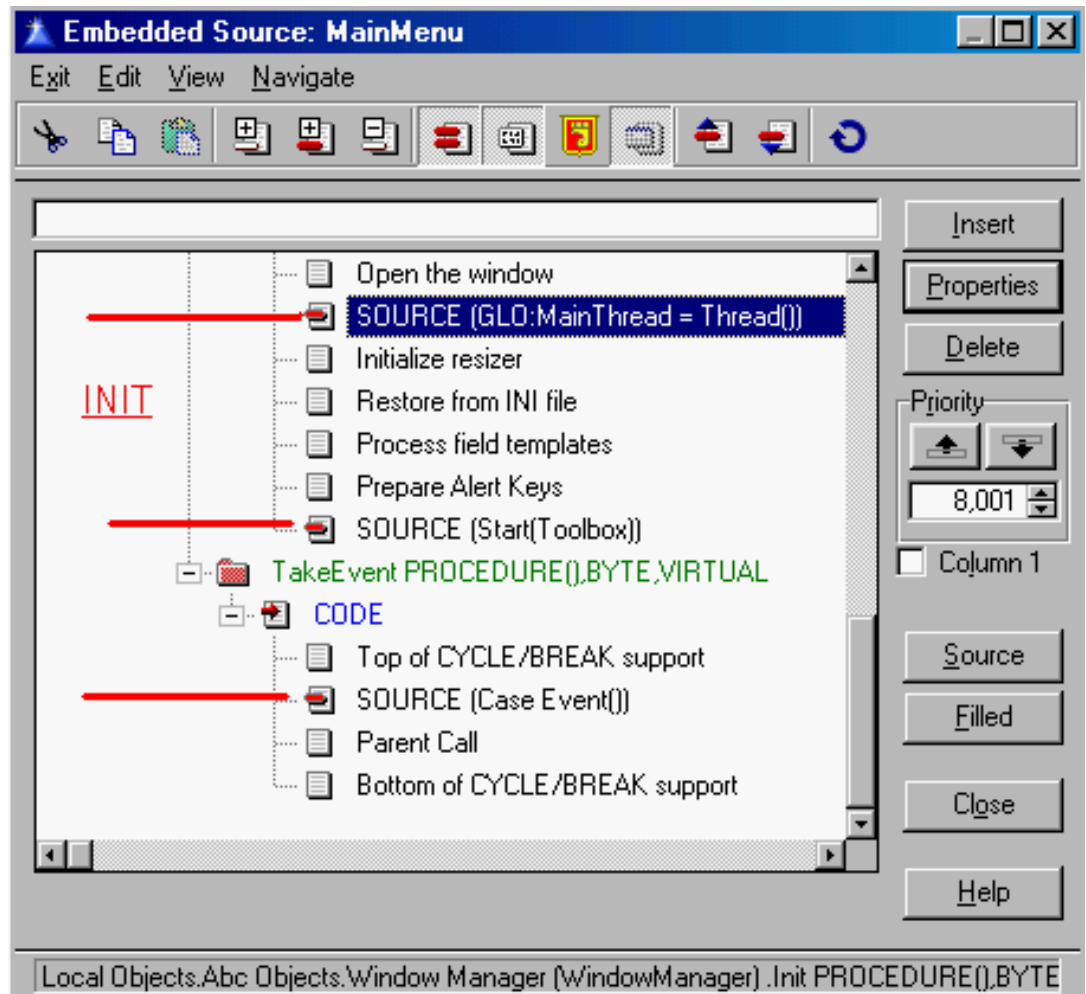
Figure 4. Required Global data.

```

Before File Declarations (ARBA.CLW)
Exit| File Edit Search
[Icons]
Event:DisableToolBox Equate(401h)
Event:EnableToolBox Equate(402h)
GLO:MainThread Byte
GLO:ToolBoxThread Byte
    
```

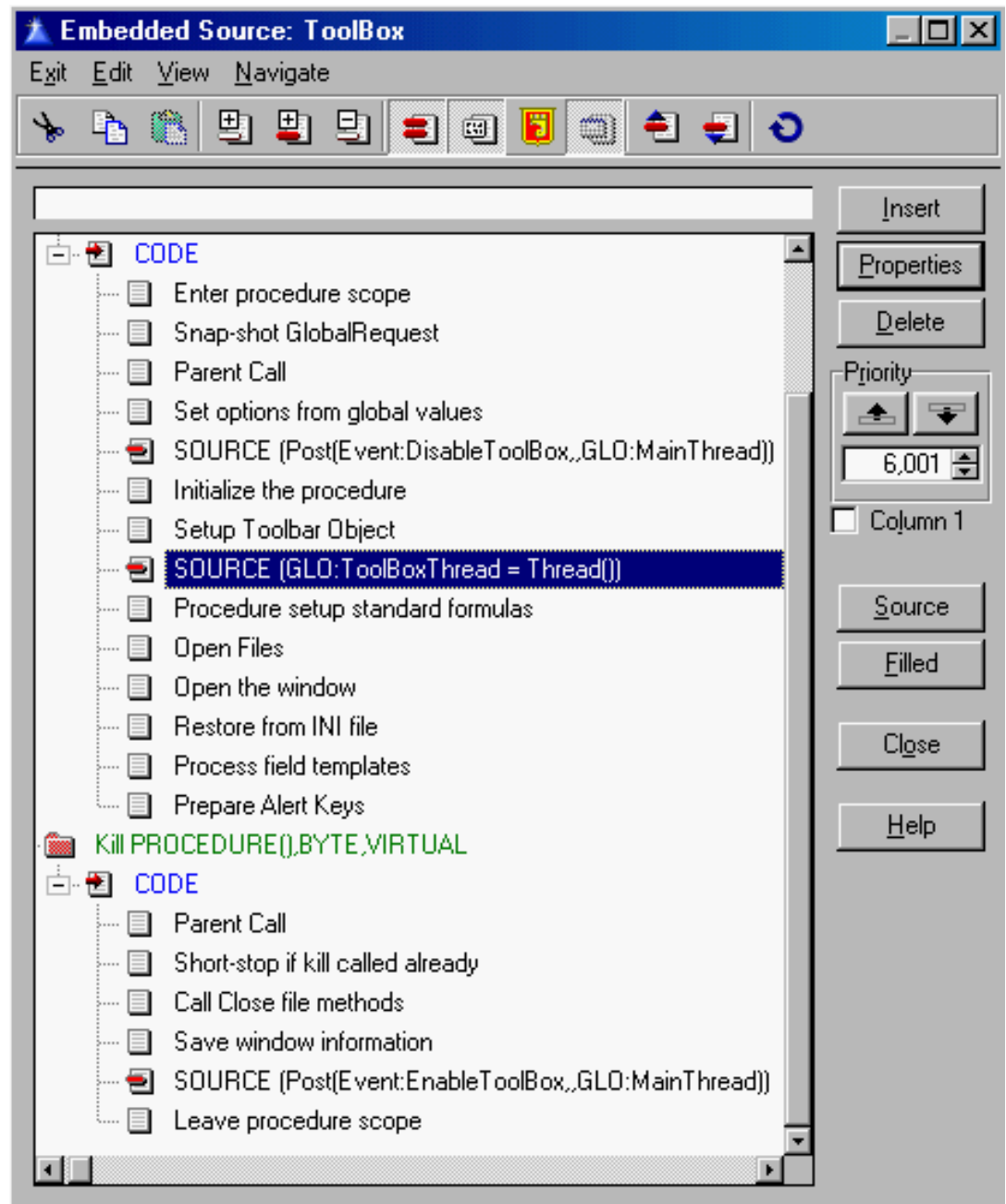
In the frame's INIT method, I capture the Thread() and Start() the toolbox.

Figure 5. Initializing toolbox operations.



Similarly, when the toolbox starts (*whenever* it starts), I get its thread:

Figure 6. Toolbox Thread () capture.



The final item is adding two items to the frame menu to open and close the toolbox. These, I place these below a separator on the standard "Window" menu.

The Open Toolbox menu item simply `Start ()`s the toolbox (and gets its `Thread ()`). This is just a standard procedure call.

The Close menu item cannot directly close the window because the menu window is on a different thread. But, because I *do* know the thread and the desired event, I can:

```
Post(Event:CloseWindow,,GLO:ToolBoxThread)
```

So, now the frame's menu can open and close the vertical menu.

Syncing The Menu

If the toolbox is open, I do want the Close Toolbox menu available but I do not want the

Open menu available. Similarly, if the toolbox is closed, I want the Open menu available but not the Close menu. It makes no sense to open what's open or close what's closed, does it?

This is where the user defined events (equates) come in. They will tell me what the user did and, from that, I can figure out what I need to do.

When the toolbox window opens, I want to disable the Open menu and enable the Close menu. So, knowing the frame's `Thread()` and, therefore, where the main menu is, I can advise the frame that the toolbox has opened and, therefore, the "Open" menu is to be disabled:

```
Post(Event:DisableToolBox,,GLO:MainThread)
```

And when the toolbox is closed, I want to enable the "Open" menu and disable the "Close" menu:

```
Post(Event:EnableToolBox,,GLO:MainThread)
```

(Admittedly, my event labels are not quite as intuitive as they could be.)

The embeds are `INIT` and `KILL`, respectively (see Figure 6, above).

Now, the toolbox is advising the frame when it is opening and when it is closing. It only remains for the frame to use that information to enable and disable the appropriate menu items.

In Figure 5, above, you will notice I have code in the `TakeEvent` embed. What more logical place to act on events posted to the frame?—I *am* posting events to that frame, after all.

Figure 7. TakeEvent code to turn menu items on and off.

```
Case Event()
Of Event:DisableToolBox
  Disable(?WindowShowToolbox)
  Enable(?CloseToolBox)
Of Event:EnableToolBox
  Enable(?WindowShowToolbox)
  Disable(?CloseToolBox)
End
```

The article in the Help file does not mention including the `Case` structure. The templates have changed since that article was written (Clarion 4 was king at the time) and, should you follow the instructions exactly, the compiler will quite happily advise you that you have erred, sinning grievously.

Summary

Necessity, it is said, is the mother of invention. When someone takes one of your windows and blows it up to the size of Montana (or Morocco), necessity rears its ugly head.

I wish I could take credit for creating this technique but it's been right there in the online help for several years. The hard part was finding the correct "How Do I ..." title and, even that, someone had to help me with.

[Steve Parker](#) started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitor's right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)**TopSpeed****Clarion 5**
by TopSpeed

FREE Microsoft Internet Explorer

etc2000
EVENT SPONSOR

The Clarion Advisor: Memory Leaks And Virtual Methods

by Jeff Slarve

The other day I had a memory leak, and I was pulling my hair out trying to figure out where it was coming from (the leak, not the hair). I had a parent class and a child class, and the child class had some ANY variables which, as you probably know, can be used to store any simple data type. Of course after you're finished with the ANY variable, you have to set it to NULL to avoid leaking memory, as in:

```
MyAnyVar &= NULL
```

So far, so good. I was doing this in the child class's automatic destructor, only the code was never being called. The reason for that is I was casting the child class to a parent class reference:

```
ParentClassRef &= new ChildClassInstance
```

This is a completely legal thing to do, and useful, too, since you can have generic code that works with any derived classes. Just remember that when you're using a cast reference your code will only be able to "see" the methods and attributes of the parent.

And that was the problem. When I DISPOSEd ParentClassRef, the automatic destructor for the child was never called. The runtime thought it was looking at the parent object, which didn't have a destructor.

The solution was to put the VIRTUAL attribute on the DESTRUCT method and add the method to the parent as well. That way on a DISPOSE the parent called the child DESTRUCT, and the ANY references were cleaned up.

[Product Review: View Wizard 1.0 From Nice Touch Solutions](#)
(Jul 11,2000)[Sidebar Menus](#)
(Jul 11,2000)[The Clarion Advisor: Memory Leaks And Virtual Methods](#)
(Jul 11,2000)[July 2000 News](#)
(Jul 11,2000)

Developer
PLUS

Your source for
development tools
and add-ons.

Your outlet for
application sales.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source](#)
[Project](#)
[Issues in](#)
[PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)

Using API Threads

by Jim Kane

Clarion does an excellent job of managing API threads for programmers. As a result you get almost as much threading flexibility as Windows C/C++ programmers do, but without all the issues and difficulties they face. Clarion does such a good job most Clarion programmers are not even aware that API threads can be difficult to use.

So Who Needs API Threads?

If Clarion's so good at threading, why bother with API threads at all? Consider the program in Figure 1. When the application frame opens it automatically STARTs the ThreadA and ThreadB procedures on their own threads. Both threads share three global variables: A, B and C. Each thread calculates a different value for C, and if one thread changes the value to something the other thread isn't expecting, the procedure will terminate with a message.

In Clarion, however, this application could run virtually forever and you would not see a message indicating the test failed in either ThreadA or ThreadB. Clarion ensures only one thread executes at a time and one thread only yields to another when an accept loop is running.

Figure 1. Testing thread management.

```

Program
Map
  ThreadA()
  ThreadB()
End
A long,auto
B long,auto
C Long,auto
Window WINDOW('Test'),AT(,,145,46),SYSTEM,GRAY,DOUBLE
      PROMPT('Running...'),AT(59,18),USE(?Prompt1)
      END
Code
Open(window)
Accept
  If event()=event:openwindow then

```

[Using API Threads Part 1](#)
(Jul 18,2000)[A Tale Of Three Brokers](#)
(Jul 18,2000)[July 2000 News](#)
(Jul 18,2000)



Your source for
development tools
and add-ons.

Your outlet for
application sales.

```

    Start(ThreadA)
    Start(ThreadB)
  End
End
Close(window)

ThreadA procedure()

Window window('dummy'),timer(1)
end

  code
  open(window)
  0{prop:hide}=1
  accept
    if event()=Event:Timer then
      Loop 10 times
        A=10
        B=20
        C=A+B
        If C<>30 then post(event:closewindow).
      End
    end
  end
  Close(window)
  Message('Thread A ended')

ThreadB procedure()

Window window('dummy'),timer(1)
end

  code
  open(window)
  0{prop:hide}=1
  accept
    if event()=Event:Timer then
      Loop 10 times
        A=20
        B=40
        C=A+B
        If C<>60 then post(event:closewindow).
      End
    end
  end
  Close(window)
  Message('Thread B ended')

```

Yet the same type of code written with API threads would bomb very fast! Why? Because in windows, a switch between threads can happen at any time. After one thread sets the value of A, a thread switch may happen and the other thread may be assuming A has a different value. Run the code in the [source zip](#) called BadThread.exe to watch a very similar API code fail on

any 32 bit OS.

Why API Threads

So why bother trying to create API threads and risk these problems? There are a number of reasons. When a Clarion program needs to wait for something to happen or a process to complete, the standard approach is to start a Clarion thread and create a window with a timer. In `Event:Timer` you can check to see if the something happened. The drawbacks to that solution are several: the window has to stay open during the entire wait, you may need multiple windows, and it uses processor overhead while checking constantly to see if the something happened.

If you need to wait for 100 somethings, you may need 100 windows or a way to tell an existing window to wait for another something. If the event you are waiting for happens infrequently, one way to minimize wasted resources is to increase the timer interval. Unfortunately many events require a quick response, even if they happen infrequently. This is especially true in some types of communications programs. Often the data must be detected fast and read fast to prevent a buffer overflow. In those cases, it is not appropriate to set a long timer interval.

Long calculations are another problem area. Say for example you need to loop through an array of 10000 strings to find just one. The most obvious approach is shown in Figure 2.

Figure 2. An obvious approach to long calculations.

```
Loop I = 1 to 10000
  If upper(AnArray[I])='WHATIWANT' then break.
end
```

But if you do that, window events do not get processed, and the program appears dead to the user until the loop is completed. Once again, for Clarion programmers, timers come to the rescue. Typical code involves looping through a few array elements in each timer event.

Figure 3. Using a timer to break up a long calculation.

```
Open(window)
Display()
I=1
Accept
  Case event()
    Of Event:Timer
      Loop 5 times
        If I>10000 or upper(AnArray[I])='WHATIWANT' then
          0{prop:timer}=0
          Break
        end
        I+=1
      end !loop
    Of Event:Accepted
      Post(Event:closewindow)
    end !case
  end !Accept
close(window)
```

By slicing the task up into small pieces you can still process window events. But it isn't the most elegant solution.

The API Thread Solution

Whether you have a long process to run or an event to wait for, it is possible to create and start an API thread that will do what you want with very minimal overhead. The operating system interrupts the thread when its time slice is up and lets other threads run.

The big advantage is other Clarion threads can run at the same time, and any Clarion windows are likely to stay responsive. But unless you are very careful, any use of global variables or global resources of any type from within the API thread(s) you create, *or* any calls into the Clarion runtime library from API threads you create, can cause GPFs or other difficult-to-debug problems.

That last phrase deserves some attention. Any code that uses multiple threads needs to be specially written to survive in a multi-thread environment. Unfortunately the Clarion runtime library is not designed to be used by multiple threads at the same time.

Calling any Clarion built-in procedure or doing any Clarion string, file or queue operations can cause trouble. Calls to Win32 APIs and some simple low level Clarion C library functions like `memcpy ()` are safe. Sometimes it can be a little difficult to decide what may be safe or not. The easiest way to tell for sure is to write the code then look at it in the Clarion debugger in the assembler mode. If in the disassembled code there are any CALL instructions there is probably a problem.

Fortunately it's usually pretty easy to avoid calling the Clarion runtime. Instead, you can post a message to a Clarion thread that a particular event has happened, and then do the real work back on the Clarion side. There is an example of this below.

Show me the code!

Enough of all this talk and theory. How about some code that solves a real life problem? I thought you'd never ask. Please refer to `cnotifcl.clw` in the [downloadable zip](#) as you progress through the explanation.

In a file transfer application there are times when a program has to wait for a file to appear. In one case I was monitoring for FTP uploads from any one of 88 branch offices. Each branch office had its own subdirectory. I needed an efficient way to detect if a new file appeared in any one of 88 different subdirectories. This would normally only happen one time per week for each branch. So I started a Clarion thread, added a timer and looped through 88 directories calling the `directory()` function for each of the 88 directories – right? Wrong! First of all, the chances of me spelling 88 directory names correctly or maintaining the correct branch directory name was about zero. Fortunately I found an API that would do it all with one function call. The `FindFirstChangeNotification` function will monitor a directory, and optionally subdirectories, for any one of a variety of changes, including when a file is renamed, deleted, updated, or has its attributes changed.

Figure 4. Looking for a changed file.


```

!intiate change notification (cnotifcl.clw, init method):
SELF.waitstruct.NotifHandle= findFirstchangeNotification
    (pTargetDir,pWatchSubTree,pnotifType)
If SELF.waitstruct.NotifHandle=Invalid_handle_value then
    SELF.Kill()
    return return:fatal
end

```

pTargetDir is a cstring containing the parent directory of the tree to monitor

pWatchSubTree is set to True to watch subdirectories of pTargetDir, False to only watch pTargetdir

pNotifType is any one of about 10 API defined constants of what to watch for. In this case you wanted FILE_NOTIFY_CHANGE_CREATION (40H) to look for file creations.

Getting A Handle

So that's it – all done, right? Not a chance! As you can see FindFirstChangeNotification returns a handle. A handle on a coffee cup I understand, but what is this handle? As it turns out, that handle falls into the broad category of objects called waitable objects. When a file is created the handle changes state from it's initial reset state to a signaled state. To detect the change of state you use another API call, WaitForSingleObject, and pass it the handle. When a file is created, and the handle changes to signaled, the WaitForSingleObject function will terminate, and any code placed after it (such as the message() function shown below) will trigger. If used with the equate of infinite, WaitForSingleObject will wait forever for the handle to change states. Figure 5 shows the sample code.

Figure 5. Waiting forever for a notification.

```

infinite      equate(-1)

...

If WaitForSingleObject(SELF.WaitStruct.NotifHandle, Infinite)
    Message('Wait handle is signaled - a file was created')
End

```

That's all that needs to be done, right? Just put the above into a Clarion app and see what happens – then offer me a large consulting fee to fix the new problem! The entire Clarion app stops responding until the wait handle gets signaled and the message appears. In other words WaitForSingleObject() acts like a tight loop constantly waiting for the handle to become signaled. The Win32 documentation says it waits very efficiently with minimal consumption of resources, but the side effect of the entire app going dead is not typically acceptable.

So how can you allow the Clarion threads to continue execution while WaitForSingleObject is doing it's efficient waiting act? Well, I didn't name this epic API threads for nothing! [Next week](#) I'll show you how to use an API thread to solve this problem.

[Download the source](#)

[Jim Kane](#) was not born anywhere near a log cabin. In fact he was born in New York City. After attending college at New York University, he went on to dental school at Harvard University. Troubled by vast numbers of unpaid bills, he accepted a U.S. Air Force Scholarship for dental school, and after graduating served in the US Air Force. He is now retired from the Air Force and writing software for [ProDoc Inc.](#), developer of legal document automation systems. In his spare time, he runs a computer consulting service, Productive Software Solutions. He is married to the former Jane Callahan of Cando, North Dakota. Jim and Jane have two children, Thomas and Amy.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)**TopSpeed****Clarion 5**
by TopSpeed

A Tale of Three Brokers

Steve Parker

It is quite surprising to look back and see that only three years ago, there was only one broker. Now there are three and confusion reigns as the product gains momentum.

In order of introduction, these brokers are:

- the EXE broker
- the DLL broker
- the "linked in" broker

The differences between them are important to understand, especially the differences in how each is deployed. Of course, most important is understanding why you need a broker at all.

Why A Broker?

HTML doesn't need a broker; CGI doesn't need a broker; ASP doesn't need a broker (well, actually, it does require additional software, similar in concept to a broker, to translate the scripting language into HTML output). So why does Clarion? (And, for that matter, why does/did Cold Fusion? And why doesn't anyone complain about *that*?)

The short story is that the broker translates Clarion screen output into HTML. HTML is what a web browser can display.

A web server, such as IIS, O'Reilly's or Apache, receives requests across the Internet. If a request directed to a web server is one it can respond to (HTTP, FTP, etc.), the web server routes the request to the appropriate service. Because more than one request can be in process at any given time, a web server ensures that responses are routed back to the correct requester.

HTTP requests are requests for HTML services and are typically configured for port 80 (21 is the default for FTP; there is no other particular significance to these numbers).

Since web-enabled Clarion apps are intended to be responses to HTTP requests, something has to convert the standard output of a Clarion app to HTML. That's the broker. If the broker didn't do it, you'd have to, using CGI (and based on Skip Williams presentation at ETC2000, this is not particularly difficult - now that *he's* figured it out -

[Using API Threads Part 1](#)
(Jul 18,2000)[A Tale Of Three Brokers](#)
(Jul 18,2000)[July 2000 News](#)
(Jul 18,2000)

Developer PLUS

Your source for
development tools
and add-ons.

Your outlet for
application sales.

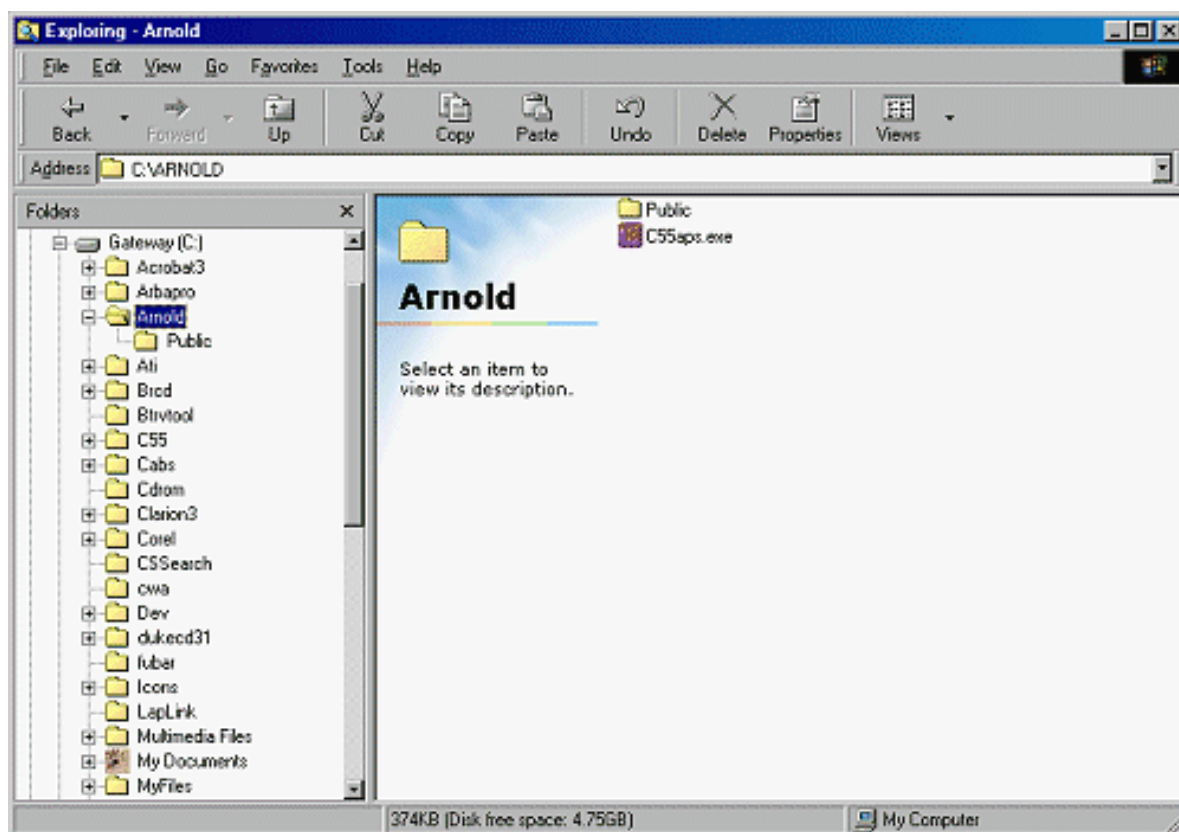
but it's not especially fun either). Furthermore, in high volume environments, CGI has its price too: each CGI request spawns processes which eat into resources; in very high volume environments, resources can be exhausted.

The EXE Broker

The first of the line, this broker is installed as an executable program (C55APS.EXE, in the unlimited version, C55APS1.EXE in the single-user/test version, C55APS10.EXE in the most current 10 user version).

The EXE broker is extremely easy to deploy. It is placed into a directory you select. The installation program defaults to \CWICWEB but *any* directory will do.

Figure 1. The broker configured in a non-default directory.



Your app(s) and *any* DLLs required by the apps also go in this directory (or directories below this one - the EXE and DLLs must either be in the same directory or the DLLs must be on your path).

If you use the installation program, a subdirectory, PUBLIC, will be created below the broker's directory. If you install manually, you will need to create this directory. CLARION.CAB and CLARION.ZIP files should be in this PUBLIC directory (if you intended to use the Internet Connect templates).

This PUBLIC directory is where any HTML and images used by your applications should be deployed (or directories below PUBLIC).

In fact, PUBLIC is the virtual root of your web; that is, PUBLIC is where apps are launched.

To use the Web Builder templates, you will also need a directory for the skeletons.

Typically, this will be below the app's directory (but, as discussed in a [previous article](#), your skeletons can be anywhere the EXE can see them).

The simplest thing is to make efficient use of your existing resources.

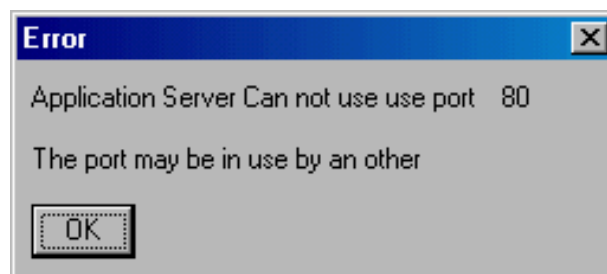
During the installation of C55, a directory call DISTRIB is created below your Clarion 5.5 directory. It contains the DLLs usually required by Internet Connect *and* Web Builder applications (you will find the TPS driver there, and if you are not using TPS files you will need to copy the appropriate file driver DLL). DISTRIB has two subdirectories, PUBLIC and SKELETON, containing the files needed for both Internet Connect and Web Builder apps.

So the easy thing is to select the DLLs in DISTRIB and the two subdirectories and copy them to the broker's (or EXE's) directory.

And you are done. No rocket science here, just copying.

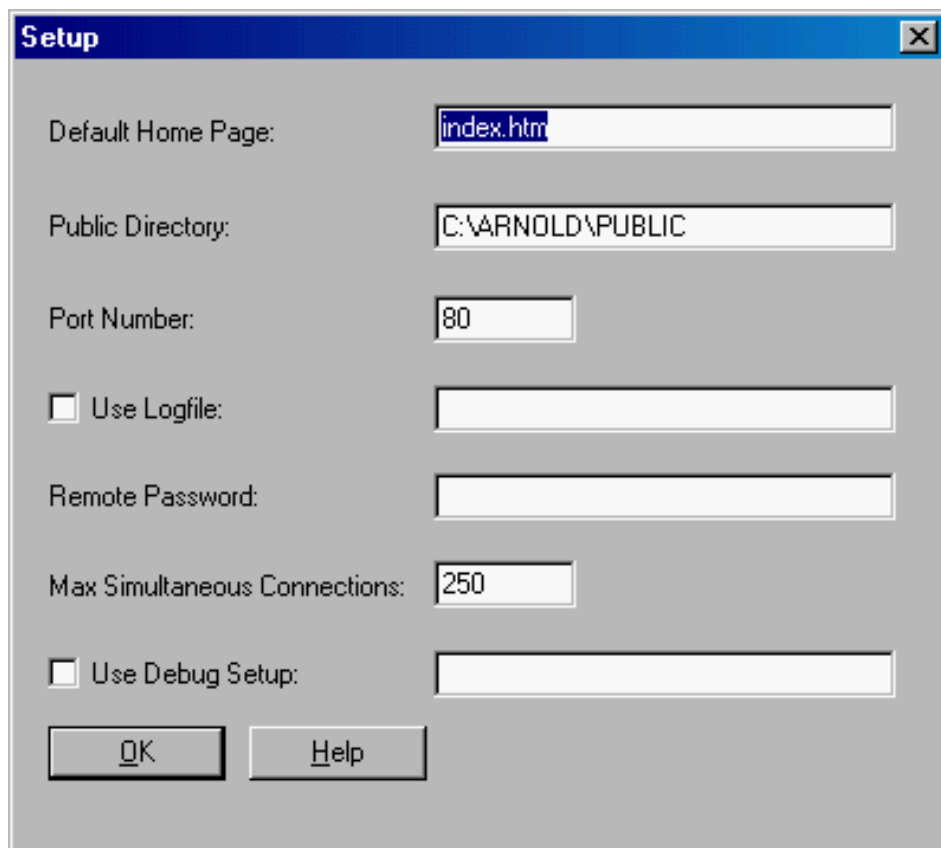
A major advantage of the EXE broker is that it can also function as a web server. That is, it does not require IIS, Apache or any other web server software. However, if you are running another web server, that web server will be bound to port 80 and the broker will also try to bind itself to port 80 on start up.

Figure 2. Broker trying to bind to port 80.



The broker will start despite this error. Click on the broker's icon in the tray, select Setup and reconfigure the port:

Figure 3. Broker configuration.



Then stop and restart the broker.

The major downside to the EXE broker is that it must be started (it is not a service), so if your NT server crashes while you're away and is restarted remotely (using PCAnywhere, for example) or your W2000 server automatically reboots itself, the EXE broker won't automatically run.

A functional limitation is that the EXE broker can only service one request at a time. So requests involving large data returns may slow a system's response time. That is, while one request is being serviced, other requests are blocked. This may be a problem for large sites or sites running on limited bandwidth.

Blocking may take place. However, it has been my experience that bandwidth has a much greater impact on performance. I ran my site on this broker for several months without any noticeable performance problem, and for smaller sites and apps would seriously consider using it (there are Clarion developers who will not consider using anything else).

The DLL Broker

The DLL broker runs as an NT service and, therefore, requires IIS or an IIS compliant web server. It *never* requires an alternate port. There are Clarion users running Clarion Internet applications on both O'Reilly's and Apache for NT successfully.

Because the DLL broker runs as a service, in the event of a system crash and reboot, the broker automatically starts up, even before a user log-on. Therefore, unattended restarts are accommodated (as, therefore, are Windows 2000's automatic restarts).

Another advantage of the DLL broker is that it serves multiple requests simultaneously. It does not block incoming request. In other words, it runs faster.

The downside to the DLL broker is that installation can be difficult.

Michael Brooks has written an extensive tutorial on installing the DLL broker, with pictures. It is available at www.clarionet.com and, if you are planning on installing the DLL broker, I recommend you check out this article.

The important thing to keep in mind is that you must make settings for both the local system and for the outside world (i.e., visitors to your site). All users on an NT system have to be identified and, by default, visitors on the Internet are identified by the Internet Guest Account. You need to configure this account correctly and precisely.

The DLL (ISAPI) Broker: Installation (NT)

(assuming IIS and the broker are installed)

- Start|Programs|Microsoft Internet Information Services|Internet Server Manager
- Expand the Default Web Site node and right click
- Select New|Virtual
- Set the Alias for CWPUB (i.e., enter "cwpub")
- Enter/select the hard path to your PUBLIC (PUBLIC is your web's virtual root)
- Specify access right per the docs (Read Only)
- **Repeat** for the remaining directories (CWSEC, CWS, CWSS)
- **Set up the Internet Guest Account for Write and Delete privileges:**
- Start NT Explorer and right click on your PUBLIC
- Select the Security tab
- Select Permissions
- Add Internet Guest Account to the User List
- Press ADD
- Press Show Users
- Select the Internet Guest Account
- Select Special Access using the "Type of Access" drop box
- Check the rights needed and click your way back out.
- **Repeat** as necessary

The default installation will create a directory called CWICWEB as well as EXEC, PUBLIC, SCRIPTS, SEC_SCR and SECURE directories below it (there will also be an ADMIN directory containing a local configuration utility). Of course, you can name another directory in place of CWICWEB. In this scenario, PUBLIC is your virtual root and is the directory you will declare as the web root in IIS.

The installation will place the required DLLs where they need to be (C5ISAPI.DLL will go into your SYSTEM32 directory; other than that, all DLLs are in the directories named above). You place your app and the DLLs it requires in EXEC or directories below EXEC; any HTML or images needed go in PUBLIC or directories below PUBLIC.

And, if you are using an SQL database, check out [Troy Sorzano's article](#) at for some tips on configuring ODBC.

Maintenance

Both brokers can be maintained remotely (except for password entry and change) by typing domainName/cws/c5launch.dll/appbroker/.

Figure 4. Broker maintenance page.

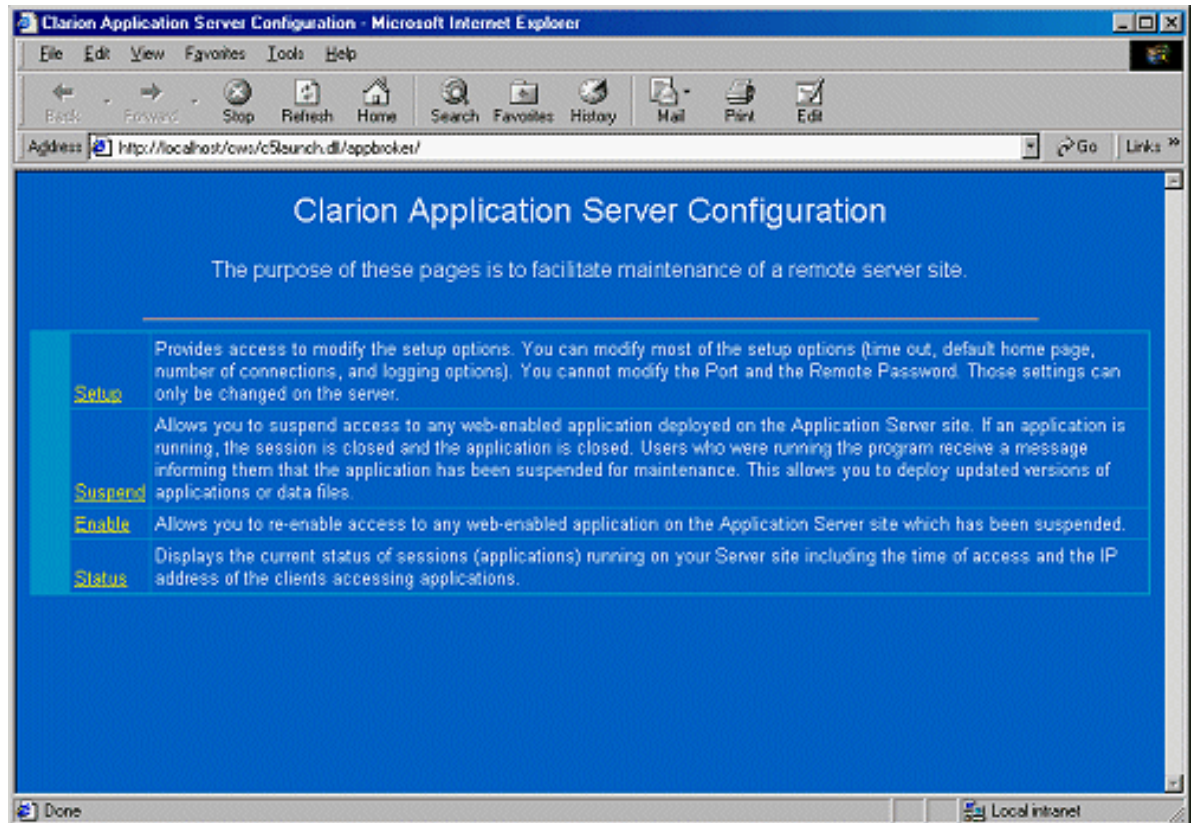
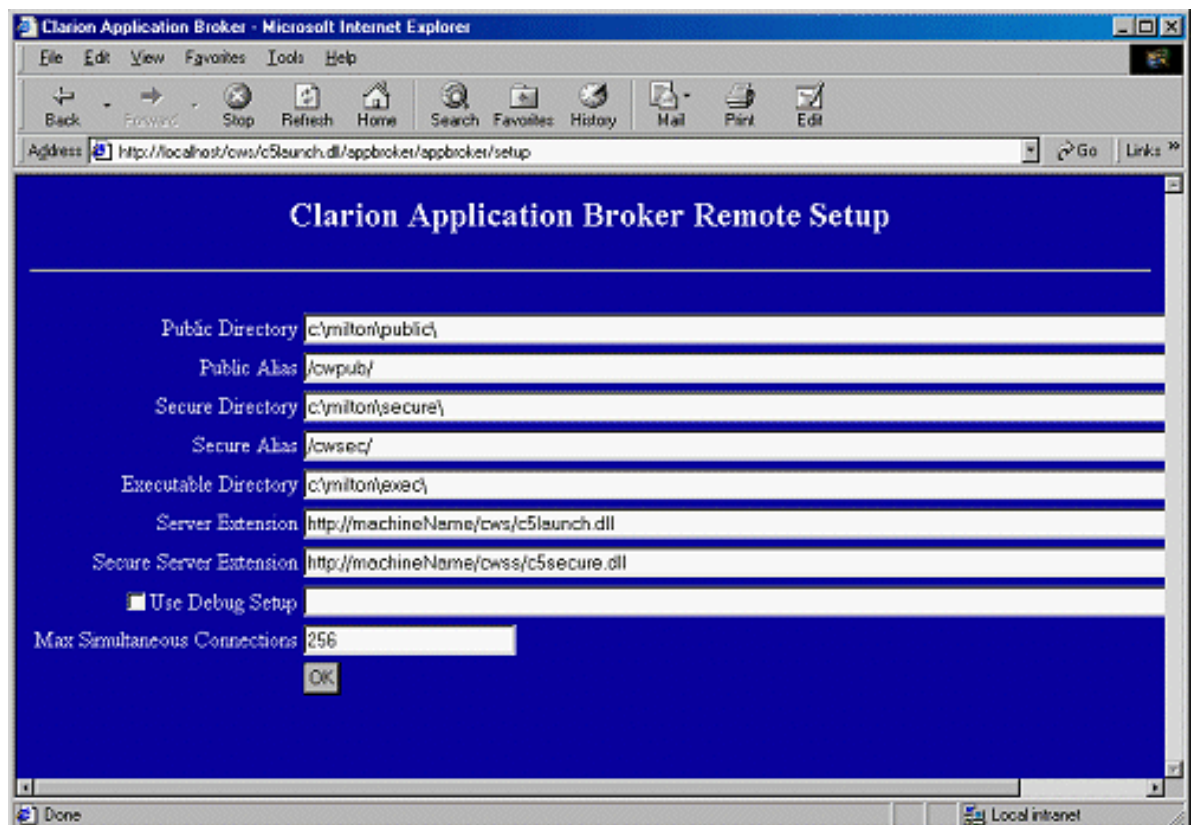


Figure 5. Broker configuration (DLL broker).



The "Suspend" command allows you to terminate any running instances of an application and prevent new visitors from accessing it. Always suspend before uploading new data files or revised EXEs. If someone is using the app when you attempt to update files, you will be prevented from loading them.

"Enable" undoes a suspend. It's good form to do this if you've previously suspended an app for maintenance.

"Status" shows which apps are running and permits selective termination of a single instance of an app.

The Linked-in Broker

Latest and greatest from SoftVelocity, the linked in broker eliminates one of the great hassles of testing apps on the web. The linked in broker allows in-line testing by simply pressing the "Run" button.

Previously, developers had to have a locally accessible web site on which to test. This is the reason that the EXE and DLL brokers existed in single user and unlimited versions; you tested using the single user version which was in the box with the templates. You'd compile, copy to the local copy of your site, run the browser, run the app, go back to the IDE, make changes, compile ...

The only thing worse was root canal.

The linked-in broker is available with the Web Application extension, not the Internet Connect templates. And, if you do not make your app dual-mode, when you run your app, it will start a browser and come up in the browser whether or not you have a web server installed. But the linked in broker *is* smart enough to detect if you have another web server running and will pick up the first available port (starting with 81) and dynamically bind to it.

The linked in broker also allows creating web demos that run locally on PCs. As before, copy the DLLs in the DISTRIB directory to your EXE directory; copy the PUBLIC and SKELETON directories below. Installed on any PC with a browser (or run from a floppy) and your app will launch in a browser.

It's not quite WYSIWYG but it's awfully close.

Supported Combinations

The EXE broker works with Windows 95, 98, NT and 2000.

The DLL broker also works with these operating systems but you must have an IISAPI compliant web server for the target O/S. IIS is available for NT and 2000. PWS (Personal Web Server) version 2 (which came with FrontPage 97) will work on 95 and 98; PWS4 will work under NT. O'Reilly's, I believe, works under NT, 95 and 98 and Apache is available for NT.

So, if you can get a copy of PWS2, you can demonstrate your app from a Windows 95 or 98 server using cable modem or DSL from your office or your home.

Web Roots: Virtual and Otherwise

When you access a web site by typing a simple URL such as www.par2.com or

www.clarionmag.com, even though you do not name an HTML file, you are indeed accessing an HTML file. In the configuration of the web server, the administrator selected a name that serves as the default page. Called a "home page," it is typically named INDEX.HTM or DEFAULT.HTM.

Now the question is "How does the web server know where to find the default home page?" For example, if using the EXE broker for your entire site, you would place your home page in \CWICWEB\PUBLIC (assuming you accepted the installation default directory). The EXE broker knows to look in PUBLIC for HTML files. So, when I was using the EXE broker, when visitors typed "www.par2.com:8080," the broker knew to look in my PUBLIC directory for DEFAULT.HTM.

IIS, on the other hand, from version 4 on, is designed to run multiple web sites (that is, multiple domains/IP addresses). Therefore, when setting up a new site, such as a Clarion web site, you need to point IIS to where the home page for that site will be stored (and, note, you can configure the broker for either a domain name or an IP address). This directory is the PUBLIC directory and, from IIS' point of view, is the "virtual root."

This is done from the Internet Server Manager (see the installation instructions, above). If you name another directory, which you can, you must move all HTML and images to the directory you have selected (though its ALIAS must be "cwpub" or the broker won't know what to do – the aliases are hardcoded in the broker – see Figure 5, above). Many Clarion users incorrectly select the top directory in the tree representing their site and then wonder why they can't serve HTML (e.g., return to pages) or images. IIS thinks those files are in CWICWEB (or whatever directory is the top node of the tree). All will function correctly when the default contents of PUBLIC are copied to this directory or when the virtual root is changed to point to PUBLIC.


Summary

The linked-in broker, which raised quite a stir when first mentioned publicly, is really a convenience for testing. Having it means not having to create a local web site for testing. It means that you can get a good look at your app as it will look on the web in almost real time and it means you can do local demonstrations.

On the other hand, the linked-in broker has fooled many into thinking that they do not have to install the EXE or DLL broker to serve apps to outside visitors. Wrong. And, while I'm on the subject, no, neither broker can be installed remotely nor can either be installed without the knowledge or cooperation of your ISP. You will serve Clarion web apps from a server you control or through a Clarion-friendly hosting service like www.cwicweb.com.

Which broker is appropriate for you? Begin with the EXE broker; it's easy. If your site is light duty, stay with it. If you do not need special services or add-ins, stay with it. For full integration into existing sites or servers, move on to the DLL broker. For heavy usage or if you're dealing with large amounts of data, use the DLL broker.

[Steve Parker](#) started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitor's right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.



Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)[Links To Other Sites](#)[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)**TopSpeed****Clarion 5**
by TopSpeed

Clarion 5.5 Gold Candidate: A First Look

by Andrew Guidroz II

Well, when are we going to see Clarion 5.5 go Gold? I know, I know, everyone wants the answer to that one, including the people who make Clarion. I'll try to answer some of those questions and also give you a look at what's coming in the 5.5 Gold release.

Lots of changes both in the product and in the management of the entire Clarion line have made for the longest beta that I have been involved in my time using Clarion for Windows.

In case some of you have been living under a rock, the Clarion product line is now being maintained and enhanced by SoftVelocity, a new company headed by Bob Zaubere. Bob is a veteran of the old TopSpeed and is different from prior presidents in charge of Clarion: he actually programs in the product.

SoftVelocity has contracts with the TopSpeed Development Centre to maintain the product. This makes for some continuity. I still see the names of folks we all know, like Gordon Smith, Nigel Hicks, and Scott Ferrett. But SoftVelocity also has the freedom to expand the product and bring in anyone they choose to do development. This will allow for new directions and gives us the advantage of buying product from a company that isn't caught in that "it must be invented here" mindset. I believe this is good.

I use Clarion to develop applications for the insurance and finance industry as well as a host of other industries. Currently, I'm also developing a package for a company that sells petroleum fuel products. These applications cover many features such as remote communications, interactive voice recognition, pure database applications, and complex reporting. Most of these databases are large though not huge. I use the TopSpeed file format for many of them, although I am also currently developing a series of new applications utilizing SQL. I also use the ABC classes and templates for all of my development.

Alpha Testing 5.5

I have the opportunity to alpha test Clarion 5.5 and I tend to do it in a method a little

[Clarion 5.5 Gold Candidate: A First Look](#)
(Jul 25, 2000)[Legacy to ABC: There is Another Way!](#)
(Jul 25, 2000)[Using API Threads - Part 2](#)
(Jul 25, 2000)[July 2000 News](#)
(Jul 25, 2000)



different from most alpha testers. I attempt to address any problems in an alpha build as soon as possible, then move it into a production environment. I have always felt that testing in a lab is too sterile and not representative of what end users are capable.

The only alpha builds that remain installed on my machine are those that pass production tests. If an alpha build is not stable enough to produce production software, I tend to post bugs related to the issues I have found and fall back to a prior release. Many of my applications were started during the early days of ABC and Clarion 4, so I have a rather large code base to test with.

My biggest issues in the older versions of Clarion were related to threading and ODBC. Because of the complex things that ABC does, and due to the fact that more code is being executed for some system modal events, there is a bigger window for timing and threading problems to occur. Different file drivers can change the timing of applications also. This can cause a threading issue to materialize when a particular driver is being used.

Clarion 5.5 has been so stable that I no longer have Clarion 5.0 installed on my machine. Every bit of my Clarion production work is being done with the latest alpha build of Clarion 5.5. The applications I currently have in production are mostly TPS file based with dozens of users running on a variety of machines from Windows 95 to Windows 2000. All of these applications are 32 bit ABC, and I currently have no outstanding issues in any of my production code.

The Details

These are the areas I have seen work done in since the public release of Beta 2.

First, the file driver corrections.

My shift toward SQL via ODBC after spending years developing with TPS and ABC has brought to light other bugs that were, to my knowledge, previously unreported. SoftVelocity has made many corrections in the file drivers and in the area of threading in the runtime library. These corrections help to make Clarion and ABC more SQL (or any other ODBC backend) friendly.

MySQL requires a different ordering of ODBC API calls than other SQL backends. The ODBC driver has been updated to handle this, as reported in [Clarion Magazine](#).

There are some new language statements: `FreeState`, `RestoreState`, and `GetState`. Those of you who have been using ABC have had the advantage of using the `SaveFile` and `RestoreFile` methods of the File Manager. Those methods save all the state information about a file (including the record buffer) and allow you to restore everything at some later point. The addition of new language-level equivalents gives this same power to Legacy folks. The beauty of ABC comes up here again in that the classes are changed to take advantage of these new language calls, but your existing ABC code works just as well as before.

One of the things that really held me up was that a reset of a file using the ODBC driver could fail if a View was ordered on a field in a joined file. This has been fixed. The ABC classes use `Reset` all over the place to keep records and windows synchronized. The side effect of this fix and other fixes in this area is that SQL works more smoothly with ABC.

Callbacks to the file drivers have been added to the language. This is often described as client side triggers. I've used them to see, and then optimize, the generated Prop:SQL

statement that is about to be sent to the backend.

The file drop classes suffered from memory leaks that seem to be cleared up now.

Combo boxes had odd behavior relating to two problems. One was dealing with the ABC classes themselves while the other involved `EVENT:Accepted` not being fired when a combo box lost focus. Both problems have been resolved. Code was also added so that the "Remove Duplicates" option for the combo box template actually works.

Forms can now be called as View Only. This makes it easier for you to display a record without worrying about the end user modifying it. There are some limitations in that any child records on that form that can be updated can still be updated. A little hand code here and there is needed if you don't want any children changed.

In the past, when you had columns within a group in a list box, the separator lines between the columns only showed when there were records in the queue. If the records didn't fill the list box, the separator lines would only cover part of the list. Now the separators look the way I would expect.

The infamous tooltip GPF in 16 bit applications in Windows 2000 has been stamped out. A 16 bit Clarion application (including the IDE) could GPF on the display of a tool tip over any of the system buttons like close, maximize, etc.

While writing this, I have been told that the loss of focus bug when within the source editor has been fixed. I haven't received a copy to test yet but my source is as good a Russian source as you can ask for.

The embed tree now allows you to press the Filled button and see the Embeditor with only those methods that have code. You don't have to page through dozens of methods that you aren't overriding.

Figure 1. From dozens of pages to just a few - using the Filled button on the embed editor.

```

Browsepeople
Exit! File Edit Search

RETURN SELF.SetSort(2,Force)
END
! Parent Call
ReturnValue = PARENT.ResetSort(Force)
! End of "Browser Method Code Section"
RETURN ReturnValue

Resizer.Init PROCEDURE(BYTE AppStrategy=AppStrategy:Resize,B'

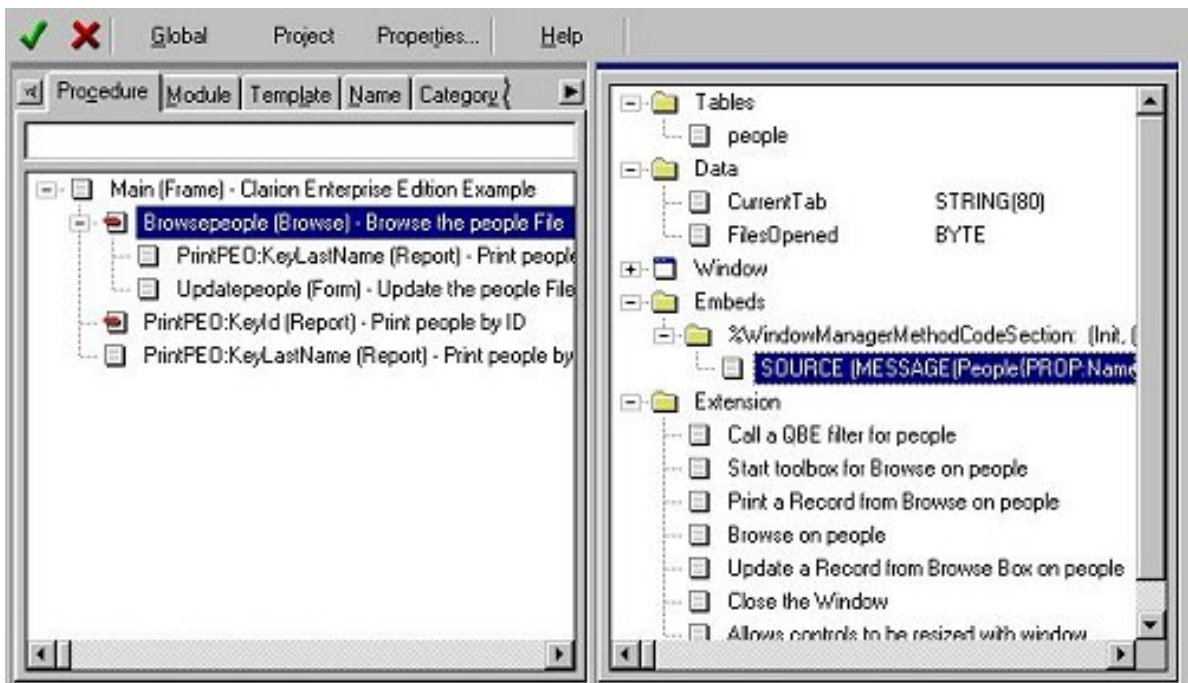
CODE
! Start of "Window Resizer Method Executable Code Section"
! Parent Call
PARENT.Init(AppStrategy,SetWindowMinSize,SetWindowMaxSize)
SELF.DeferMoves=False
SELF.SetParentDefaults
! End of "Window Resizer Method Executable Code Section"

! End of "Local Procedures"
    
```

Line: 232 Col: 1 Insert

The procedure tree now has the right hand side tree that gives you direct access to any embed point and any control or data item. This is very convenient when you want to make that quick embed change without opening the whole procedure.

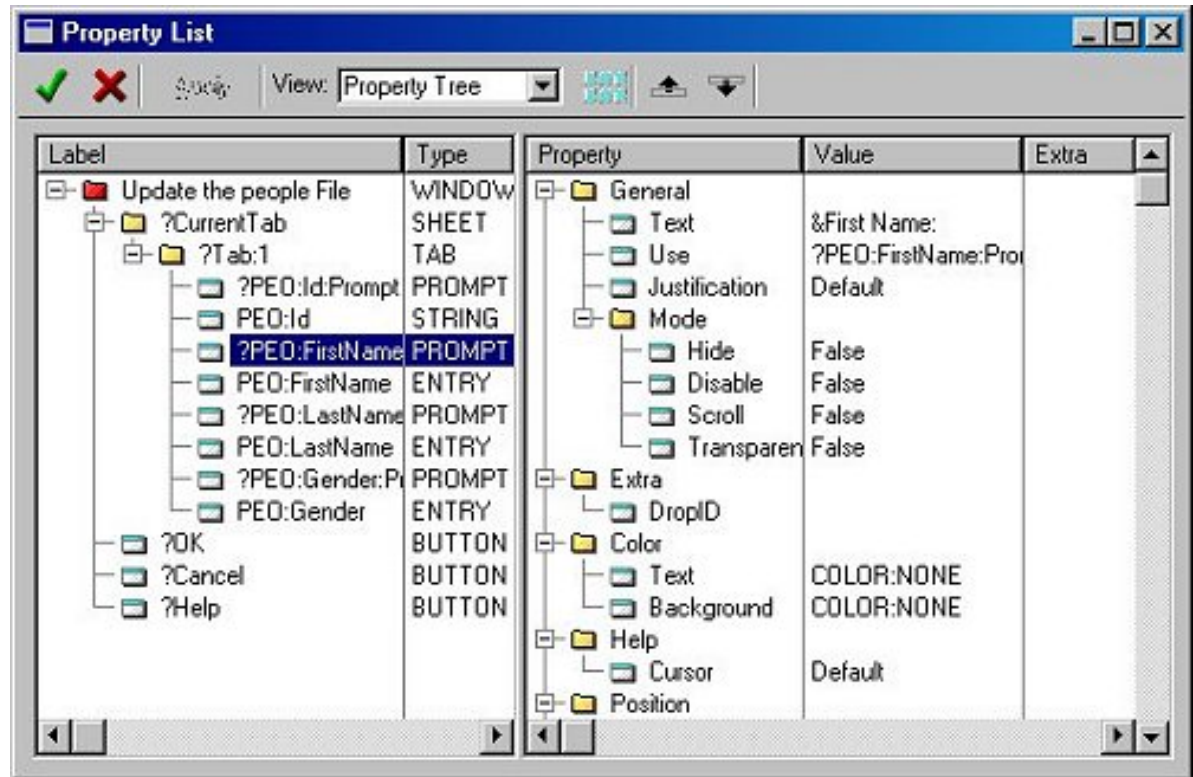
Figure 2. The split-pane procedure view



The Visual Property Editor or Property Inspector or whatever the name ends up being is

very nice. This allows you to apply changes to multiple controls at a time. This can be something as simple as adjusting a font or font size, and it can also include moving/reordering controls. It is available in the Window Formatter, the Report Formatter, and even the Dictionary Editor. Just press the F12 key and away you go.

Figure 3. The property inspector



The sorting feature contained within this editor is also very nice. Right click and select Sort>Label. This puts all the controls on your window in alphabetical order. Now, select sort again but this time by type. Now, you controls are grouped by type and alphabetical within each type. This is very nice for standardizing a look among a common group of controls.

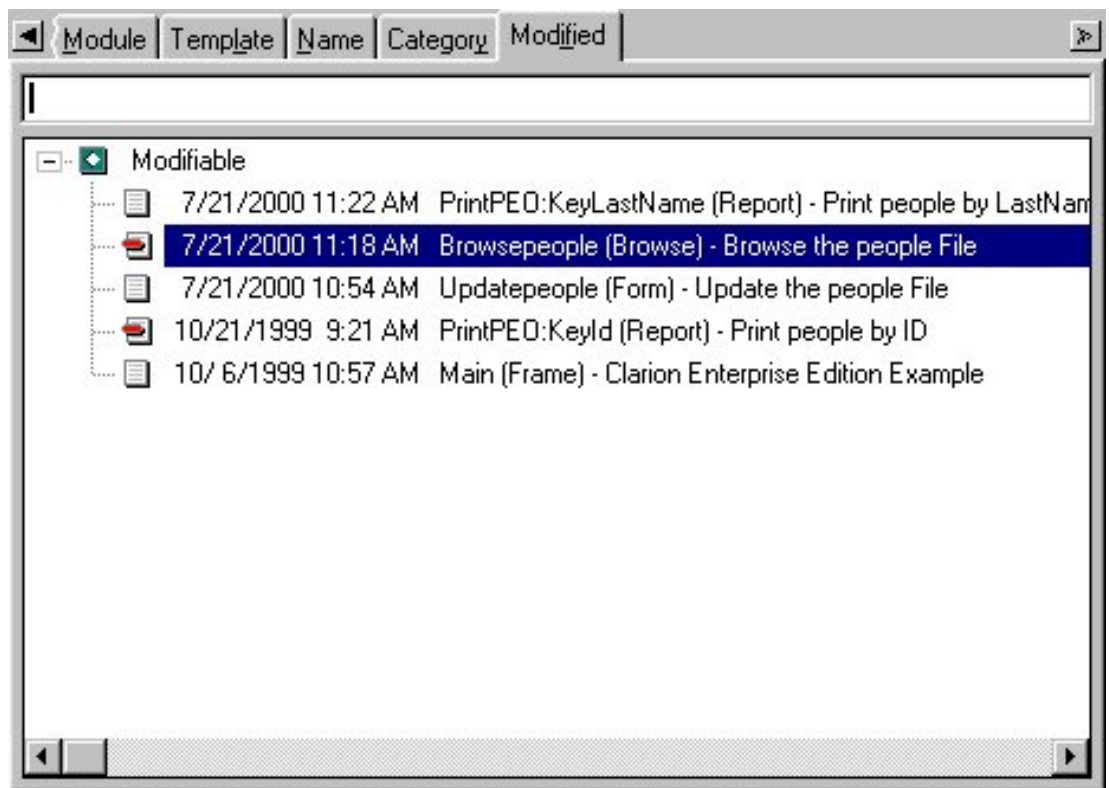
Figure 4. Sorting controls alphabetically and by type

Label	Type
?Cancel	BUTTON
?Help	BUTTON
?OK	BUTTON
PEO:FirstName	ENTRY
PEO:Gender	ENTRY
PEO:LastName	ENTRY
?PEO:FirstName:Prompt	PROMPT
?PEO:Gender:Prompt	PROMPT
?PEO:Id:Prompt	PROMPT
?PEO:LastName:Prompt	PROMPT
?CurrentTab	SHEET
PEO:Id	STRING
?Tab:1	TAB
Update the people File	WINDOW

The 32 bit debugger is much more friendly and workable in Windows 2000. I used it when I was debugging problems with the `RestoreState` function in conjunction with `Prop:SQL`.

The procedure tree has the new Modified tab available. This lists your procedures in the order they were last modified. Newly added to it is a display of the date and time a procedure was last modified. When I'm working for a client, I tend to keep Microsoft Word open and document modifications to each procedure. Sometimes I make that minor change that I forget to document and I can't determine later just what I modified in an application that session. With the time and date now showing, I can easily see that I modified the `PayrollRates` procedure just last week.

Figure 5. The modified tab on the procedure view



The Text Editor, when used to print listings out, could print very small headings depending on the default font of the printer driver being used. This has been corrected. Another new feature is the ability to save macros in the editor. This has been long requested, and I take advantage of it daily.

For some time, some people have complained about populating controls in the report formatter and having them all "jump up" to the band's upper left corner. This bug was terribly hard to recreate. Finally, someone came up with a scenario that allowed the Development Centre to recreate and fix the problem. And this brings me to an interesting point: It isn't just enough to report a bug. Step by step carefully worded directions on how to recreate bugs are *very* important. There are often many different ways to accomplish the same task in Clarion, so knowing your exact steps is critical to everyone involved in fixing bugs.

Now, it is time to attempt to answer that big question: when are we going to see it? SoftVelocity plans to distribute a Candidate Release electronically (on their web site) within the next two weeks. This is not a microwave dinner, though. You don't put together a complex package like Clarion for Windows and say, "It will be completed in 2 minutes, 32 seconds." It is more like baking bread. You can plan as much as you want as to how long it will take to prepare but the yeast, heat, and humidity have a life of their own. The product is a living organism. Given the progress made in the last few alpha builds, I think that this is a realistic goal.

I have only described work in areas that directly affected me and that probably describes less than ten percent of the fixes made in the product. This doesn't even take into account the large volume of fixes that have been made in the product before the first and second public betas.

I'm sure everyone is also interested in many of the new features that have been announced and not beta tested yet, like the Rich Text control and the new API functions. These features have been on their own development cycle and will not be released to alpha and beta testers until they have been fully documented. This policy of not releasing

features without documentation is an improvement over previous blind testing, but still frustrating because I'm as eager as everyone else is to see the new stuff.

All in all, I believe Clarion 5.5 to be a good evolutionary step forward as well as the most stable release yet.

Andrew Guidroz II is a member of Team Topspeed QA and writes software for the insurance industry. His famous Cajun cookouts have become a central feature of Clarion conferences throughout the U.S. Andrew's Cajun website is www.coonass.com.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free

CLARION
online

published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)

[COL Archive](#)

[Log In](#)
[Subscribe](#)
[Renewals](#)

[Frequently Asked Questions](#)

[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)

[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)

[Advertising](#)

[Contact Us](#)

TopSpeed

Clarion 5
by TopSpeed

FREE Microsoft Internet Explorer

etc2000
EVENT SPONSOR

Legacy to ABC: There is Another Way!

by Simon Brewer

Part 1

(Editor's Note: This article series is based on a presentation Simon gave at ConVic '99.)

Hands up if you use ABC. Great, that's most of you. Keep your hand up if you also have live Legacy (.ie. Clarion template chain) apps. Hmmm, not too many hands went down. Keep your hands up if you've more or less abandoned plans for, given up hope of, don't have time to, or are just daunted by converting those Legacy apps to ABC. What? Same number?

In my experience, the above is a fairly typical scenario amongst a group of longer-term Clarioneers. If you're still holding your hand up (by the way, you can put it down now) I'd be fairly sure you've attempted to convert a Legacy app to ABC using the tools provided, and didn't have too much luck. The project is probably shelved until a very, very rainy day, possibly one that only Noah would venture into.

The major reason is that you've discovered apps need a serious overhaul to make it in ABC. You may need to eliminate, move, modify or at least check all of your embeds, reconfigure a number of third party tools, carefully check that all of the properties you had are still set, and then test. And test. And test. All of this will only get you back to square one. Finally, you can add some of the new ABC functionality. That takes time and, as they say in the classics, *time is money!*

So, what becomes of those Legacy apps? You should know by now that Legacy templates are finished. They're no longer being developed and third party vendors are deserting them in droves. Yes, they're still there, and will continue to be supported for a while longer, but ABC is the clear direction of the Clarion development environment for many reasons. Your Legacy apps are probably somewhere in limbo as you successfully fend off requests for edit-in-place and other such niceties. It's very frustrating.

There are very good reasons for that frustration. Conversion of an average app from Legacy to ABC in one go can take as much as one hour per procedure, including thorough testing – quite possibly more when all is said and done. That means anything

[Clarion 5.5 Gold Candidate: A First Look](#)
(Jul 25, 2000)

[Legacy to ABC: There is Another Way!](#)
(Jul 25, 2000)

[Using API Threads - Part 2](#)
(Jul 25, 2000)

[July 2000 News](#)
(Jul 25, 2000)



more than a small 50-procedure app may take a week.

A more substantial 400-procedure app with a good number of third party add-ons, a few template tweaks and a healthy serving of embedded code could take eight weeks or more. Obviously that estimate varies from developer to developer, but I'm fairly sure I've hit the middle ground. The sheer enormity of the task and fear of the unknown are most of the problem – it's like deploying the app for the first time, yet nothing's changed. Well, at least you *hope* nothing's changed!

Having absorbed that, here's another chance to put your hand up. Who has the sort of time mentioned above just to get an app looking and functioning the same as it does now? What, no hands up? Actually, I'm not surprised; I don't either.

Therein lies the problem: conversions from Legacy to ABC are almost always discussed as time consuming, daunting, all-or-nothing tasks. Why? Well, my guess is that as a tool exists which appears to be an all-or-nothing converter, everyone is convinced that's the way to go. Although the converter works, SoftVelocity have acknowledged that it has limitations and given good reasons for them, so the rest is up to you. To date, viable alternatives have been few and far between, so the myth persists.

Never Give Up The Ship

However, there is hope! There is absolutely no reason to convert a Legacy application to ABC in one go. Let me say it another way: you can have one app (I'm talking in terms of a whole application rather than a single .app file here) which has a mixture of Legacy and ABC procedures in it. Best of all, it's simple, it's safe, it's efficient, and everyone can do it.

Think of the possibilities; you can gradually convert your Legacy apps to ABC in your own time, possibly learning ABC as you go. You can add new features or write new extensions entirely in ABC. You can test in small chunks with great confidence and before you know it, your entire application will be ABC. With the method I'm going to show you, you'll be able to preserve your Legacy procedures while running their ABC equivalents and even revert back if required!

To truly understand how this process works, you must know the answer to the following question: what's the difference between Legacy and ABC applications anyhow? The answer, in terms of this conversion method, is (drum roll please): absolutely nothing! Legacy is made up of global memory variables and a series of individual, self-contained procedures. ABC is the same. There is no magic to ABC.

Diagrammatically, try to envisage your Legacy and ABC applications looking like this:

Figure 1. Typical Legacy application.



Figure 2. Typical ABC application.



The differences between Legacy and ABC can be thought of as intra-procedure, not intra-application. After all, the structure of your application is the same in both Legacy and ABC, isn't it? You don't do anything different to add for example, a Form, in one or the other, do you? Any procedure created in a Legacy app to perform a particular task will be called the same, look the same and operate the same as its ABC equivalent, yet the source code generated for the two couldn't be more different. Hence most embedded code is quite different.

In line with this philosophy, many of the links between procedures in Legacy have been carried forward to ABC, to the eternal credit of the template development team. By way of example, if a Browse calls a Form to insert a record, it passes the request forward by setting a global variable (GlobalRequest) to a value (InsertRecord) before the call. The Form then knows what the Browse intends it to do. It's much harder to see ABC doing this, but it's there. Again, no magic, just sound, logical programming.

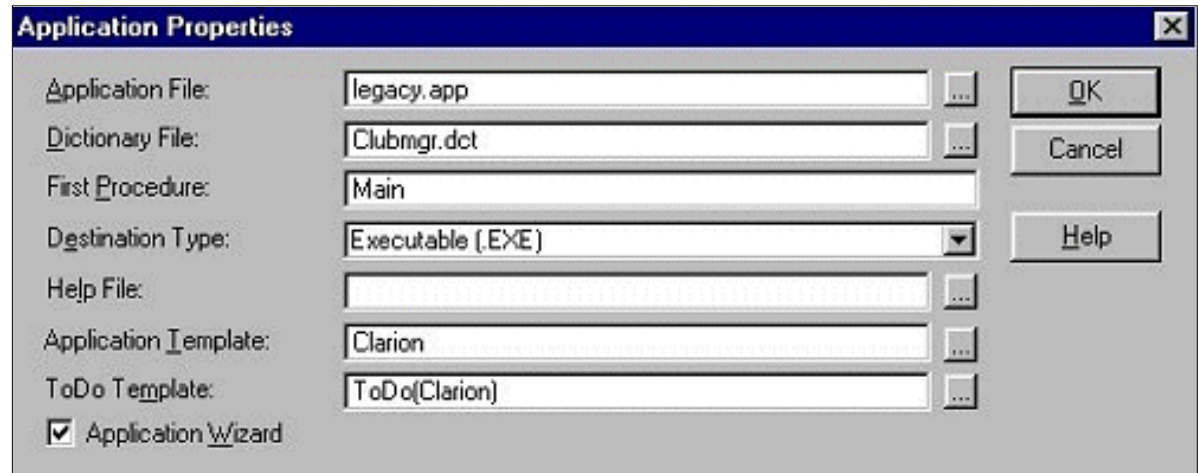
A Hybrid Tutorial

Given that background, I think the best way to demonstrate my hybrid conversion method is to take you through a step-by-step tutorial. I'll use one of the example dictionaries supplied with Clarion 5 to prove the concept so you can follow along.

To get started, create a new Legacy app using the Application Wizard and the Club Manager (ClubMgr) dictionary from the examples supplied with Clarion, just as I've

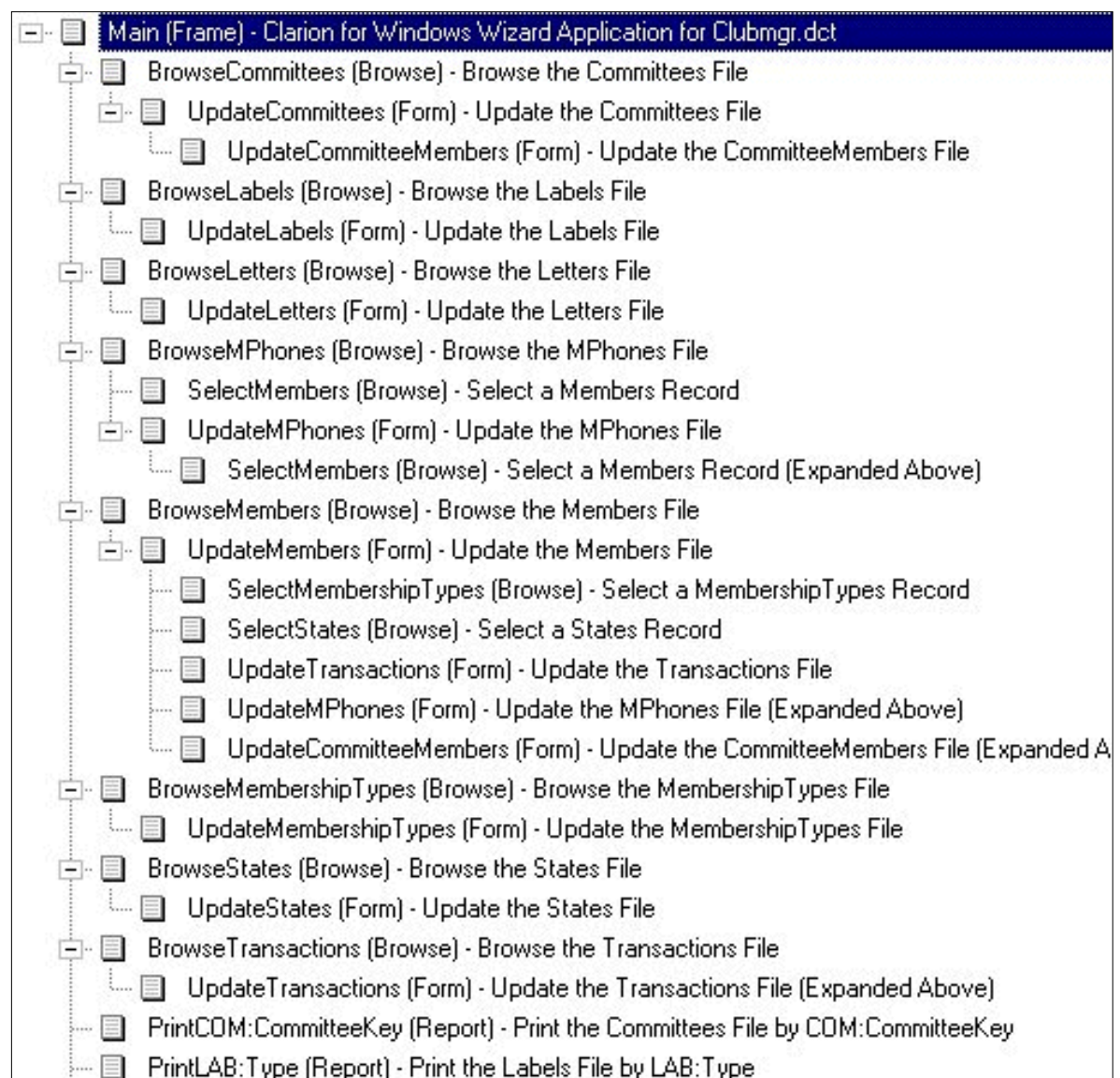
done in Figure 3.

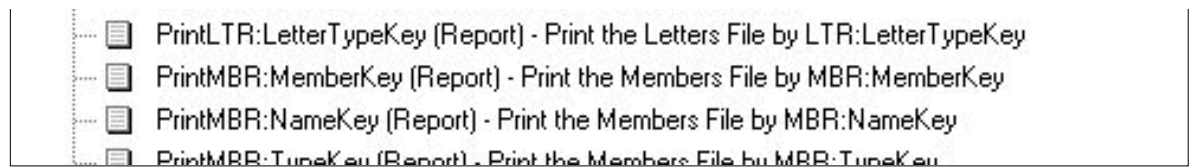
Figure 3. Creating a sample Legacy app.



Once you've done that you should end up with an app tree similar to that shown in Figure 4.

Figure 4. Tree structure of Club Manager Legacy app.





If you copy the physical database (*.tps) files across from the Club Manager example directory to your working directory you should be able to *Make and Run* the app and have real data to look at. Try that out - make sure you've got a fully working Legacy app.

Looking back to Figure 1, and taking into account what I've just told you about the differences (or lack of) between Legacy and ABC, it should be obvious that the procedures need to be isolated from the rest of the application (globals, data files etc). Extracting the orange-shaded Global Data area to a separate DLL, thus leaving the procedures, is a simple way to achieve this. Therefore, the first stage of conversion will be to create a Global Data DLL.

This DLL will contain all global data storage, including file structures, which can then be referenced from the Legacy app. However, here's the twist – this DLL will use the ABC template chain. That has the effect of not only containing the global data, but also including the ABC class libraries into the equation. It expands the orange shaded Global Data area to the size of that shown in Figure 2.

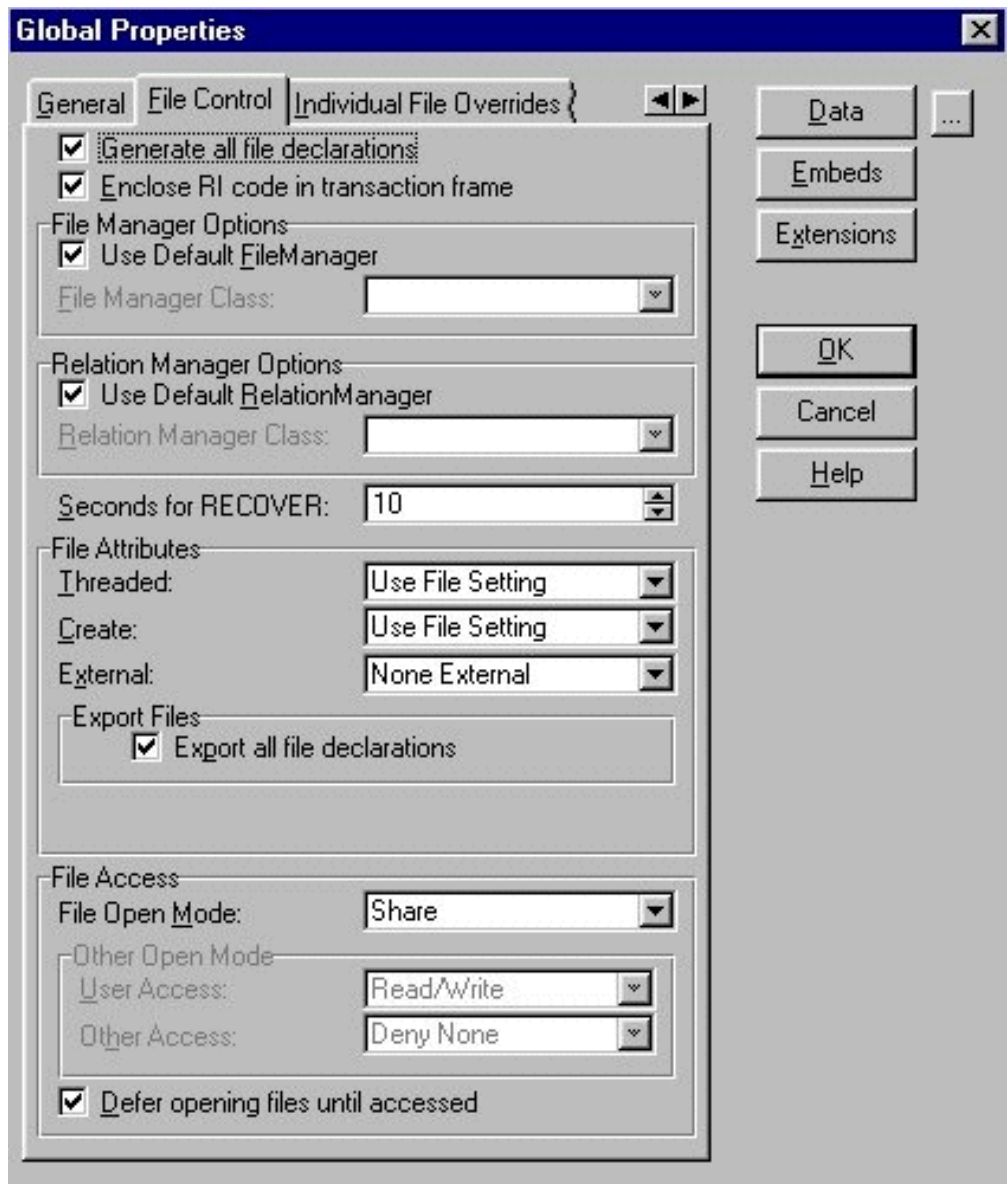
Creating DLLs may seem a difficult task if you've never done it before, however, let me assure you it's a standard operation in Clarion programming and very safe. Creating a Global Data DLL is usually the first step in breaking any application up into separate DLLs. I strongly suggest you read the appropriate sections in your Clarion manuals if you have trouble understanding what's going on below.

Creating a Global Data DLL

Create the Global Data DLL by following these steps:

1. Create an empty ABC app with the ClubMgr Dictionary but don't use the Application Wizard.
2. Set the **Destination Type** to **Dynamic Link Library (DLL)** and create the app.
3. After generation, in **Global, File Control** set the **Generate All File Declarations** and **Export All File Declarations** check boxes on (Figure 5).
4. Make the Main procedure an empty Source procedure and set the **Export Procedure** check box off on its **Procedure Properties** window.

Figure 5. Global Data section of new ABC app.



Now Make the DLL. Note that you must perform another step here if you're converting an existing app with many global data variables. That is to copy the global variable declarations across from the Legacy app to the new ABC Global Data DLL. Using copy & paste in the ellipsis (...) button to the right of the **Global Data** button on the **Global Properties** window is the best way to do that.

Mixing it into Legacy

Back in the Legacy app, a few alterations need to be made to incorporate the new ABC Global Data DLL in place of having the data stored "locally". Follow these steps:

1. Under the **Global** section, in the **General** tab, check the **Generate Template Global Data as External** check box (Figure 6).
2. Under the **File Control Flags** tab, set the **When Done With File** selection to **Keep The File Open** (more on this later).
3. In the same tab under **File Attributes**, **External** set to **All External** and check the **All Files Declared in Another APP** check box (Figure 7).
4. From the **Application** section of the main Clarion menu, select **Insert Module**, choose **External DLL** and enter the library name for the new ABC Global Data DLL you just made (Figure 8).

Figure 6. Legacy app global setting changes.

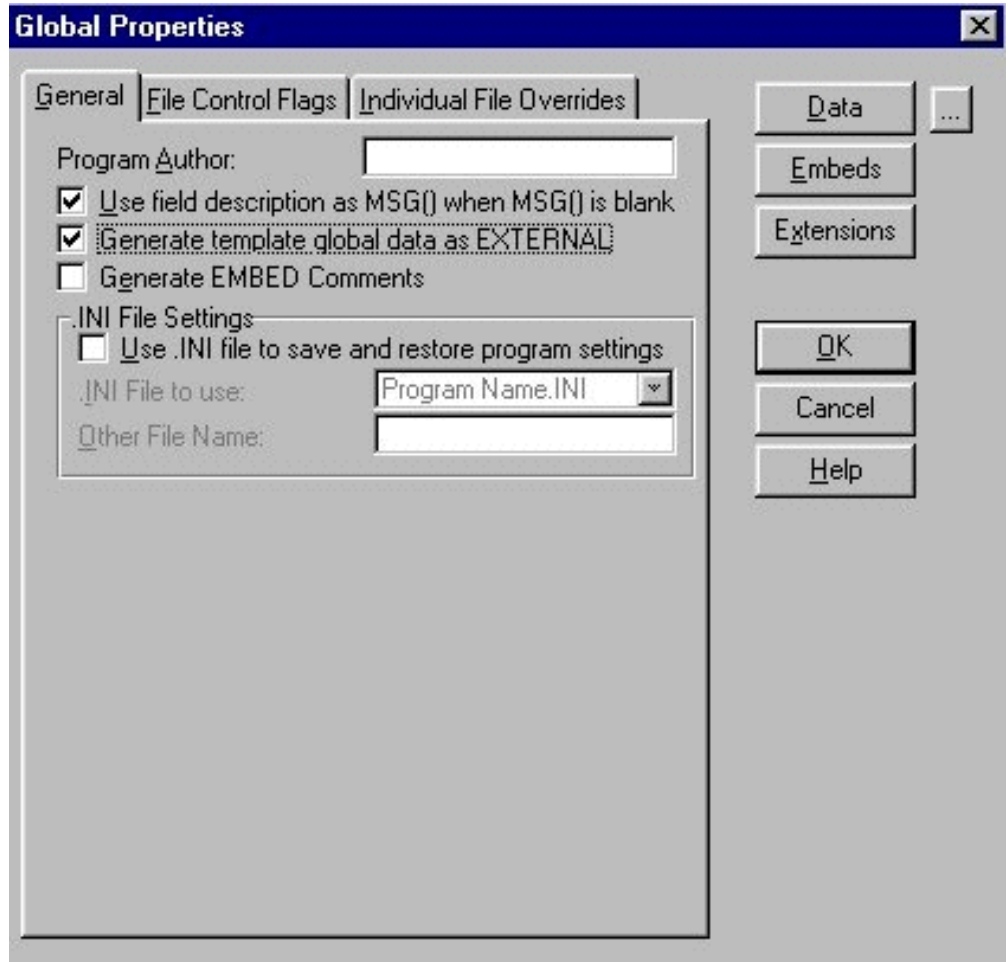


Figure 7. Legacy app global setting changes.



Figure 8. Adding ABC Global Data library to Legacy app.

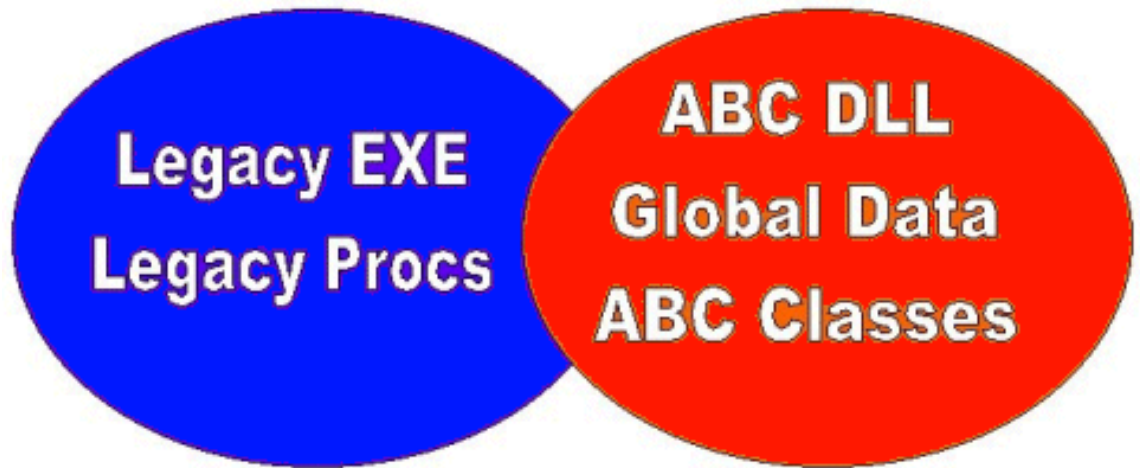


Note that if you're converting an existing Legacy app and you've copied global variables to the new Global Data DLL, you'll need to visit each global variable's properties and set the **Storage Class** on the **Attributes** tab to **External – DLL**.

Now you're ready to *Make and Run* the Legacy app. You should find that it compiles correctly and runs just fine. Congratulations, you now have a hybrid Legacy/ABC app and you've just done the major part of converting your app from Legacy to ABC! Take a bow.

Your new application can be thought of diagrammatically as looking like this:

Figure 9. Hybrid Legacy/ABC application.



The more advanced amongst you have probably already jumped to the next step, which is essentially to build upon the ABC side of the program. However, I'm trying to get you from Legacy to ABC in the easiest and safest possible way, so there are plenty more good tips to come. There's also a trap you'll need to watch out for. I'll cover all of that in Part 2 of this enthralling drama!

[Simon Brewer](#) is a Senior Analyst with Email Major Appliances, Australia's largest manufacturer of whitegoods and major Clarion users. In his spare time he is also the President of the South Australian Clarion User Group and a co-organiser of the ConVic conferences.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)[Subscribe](#)[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)[Article Index](#)[Author Index](#)[Links To](#)[Other Sites](#)[Downloads](#)[Open Source](#)[Project](#)[Issues in](#)[PDF Format](#)[Free Software](#)[Advertising](#)[Contact Us](#)**TopSpeed****Clarion 5**
by TopSpeed

Using API Threads

by Jim Kane**Part 2 of 2**

[Last week](#) I explained why API threads are important, and why Clarion's threading, though powerful, isn't appropriate for every situation. This week I'll conclude by showing how to use API threads to wait for a file to appear in a directory without making Clarion unresponsive in the process.

Creating An API Thread

The API CreateThread function is pretty straight forward. Here is the code (see the init method in notifcl.clw):

Figure 1. Calling the CreateThread function.

```

Threadid ulong,auto !required but unused
SELF.Threadhandle =
  createthread(0,| !Nt null or default security
  256,| !stack size - you need just a small one
  address(waitproc),|!procedure to start
  address(SELF.waitstruct),| !input parameters
  1,| !1=start thread immediately
  threadid) !thread id unused
If ~SELF.ThreadHandle
  messge('create thread failed')
End

```

This function call creates a new thread and starts the procedure called waitproc. The procedure that is started must have the prototype:

```
Waitproc(long), long, pascal
```

The CreateThread() procedure call passes Address(SELF.WaitStruct) to the waitproc. Since you only get to pass one long to the new procedure, lets make it the address of a group so you can pass any amount of information you want to:

[Clarion 5.5 Gold
Candidate: A First Look](#)
(Jul 25,2000)[Legacy to ABC: There is
Another Way!](#)
(Jul 25,2000)[Using API Threads - Part 2](#)
(Jul 25,2000)[July 2000 News](#)
(Jul 25,2000)

**Figure 2. The WaitStructGroupType group**

```
WaitStructGroupType Group,type
notifHandle long
hEvent      long
hwnd       long
wmsg       long
autoreset  long
end
```

The first group member is the handle from `findFirstchangeNotification`. The `hEvent` is another waitable handle that will become signaled if you want to stop waiting (more on that later). The `hwnd`, is the windows handle for the window you want to send notification to that a file arrived. `WMsg` is the message that will be sent to `hwnd`. `Autoreset` is a flag that if true causes the code to reset `FindFirstChangeNotification` and wait again after one file arrives.

The Wait Procedure

The wait procedure is pretty straight forward. You should however make a copy of the passed `WaitStruct` group immediately because you don't know how long that group will be a valid structure. The calling procedure can terminate before the API thread terminates, destroying the original group. Figure 3 shows the code (`cnotifcl.clw`, the `waitproc` procedure).

Figure 3. The WaitProc procedure

```
waitproc procedure(long lpwaitstruct)
loc:WaitStruct like(WaitStructGroupType)
infinite      equate(-1)
Wait_object_0 equate(0)
code
memcpy(address(loc:waitstruct), |
        lpwaitstruct, size(WaitStructGroupType))
loop
!if waitstruct.notifhandle has become
! signaled then send a message
if WaitForMultipleObjects(2, |
address(loc:WaitStruct.notifhandle) |
,0,infinite)=Wait_object_0 then
SendMessage(loc:waitstruct.hwnd, |
            loc:waitstruct.wmsg,0,0)
if loc:waitstruct.autoreset then
FindNextchangeNotification(|
            loc:waitstruct.notifhandle)
cycle
end
end
break
end !sleep again if reset
return 0
```

The `memcpy()` function is a portion of the Clarion run time library that it is safe to call; it copies the `waitstructure` passed to `waitproc` to local memory. The first parameter is the destination, the second is the source, and the third is the number of bytes to copy.

WaitForMultipleObjects

`WaitForMultipleObjects` is very similar to `WaitForSingleObject` except it can wait for up to 64 objects (2 in this case – `hevent` and `Notifhandle`) and if either becomes signaled, it returns the identification of the object that became signaled. In this case, since `notifhandle` comes first in the group, it is called object 0. When the file change you are waiting for happens, the notification handle becomes signaled and as a result, `WaitForMultipleObject` returns `Wait_Object_0` (or 0).

When `WaitForMultipleObject` returns `Object_0`, the next step is to call the API `SendMessage()` function to send a message to another Clarion window signaling the event. The target Clarion window (whose `hwnd` you passed in the group) needs to be [subclassed](#) to detect receipt of the message.

The class `cnotifcl` in the (downloadable zip) automatically subclasses the top window on the thread it is passed in the `init` method. The class does all the hard work. When the subclass code receives the message, it posts a specified Clarion event to the Clarion thread passed to the class `init` method.

The user of the class (that could be you!) passes a Clarion thread, Clarion event, details of what type of file change to wait for, a few flags to further customize the wait and almost by magic, the Clarion event appears when the prescribed file change occurs. After sending the message, if `autoreset` passed in the group was set to true, then `FindNextChangeNotification()` is called to rearm the process. The big advantage of all this is the API thread sleeps and waits for `WaitForMultipleObjects()` to become signaled without dragging down the Clarion application.

There's only one problem with this sleeping thread: . how do you wake it up and tell it to quit if you want to abandon the watch for file changes or close the application? Fortunately it's not too difficult since you had the foresight to put two different objects in the wait: the event object (`hevent`) passed in the group as well as the handle returned by `FindFirstChangeNotification`.

Event Objects

Event objects are quite handy. They have two states, reset or signaled. When signaled, any of the wait functions return immediately. Most commonly you create an initially reset event object, put it in a wait function, and when you want to wait function to return, cause the event object to become signaled. The code to create an initially reset event object looks like Figure 4.

Figure 4. Creating an initially reset event.

```
SELF.WaitStruct.hEvent = createevent(
    0, | !nt default security
    1, | !manual reset;
    0, | !0=initially reset; 1=initially signaled
    ) !null or no name
```

In the class `kill` method, when you want to end the wait, all you have to do is set the event

object to signaled. This causes `WaitForMultipleObjects` to return immediately which causes the `waitproc` to end, and hence the API thread to end. The code to do this is in Figure 5.

Figure 5. Signaling the end of the wait

```
!kill the alt thread
If SELF.Threadhandle and SELF.WaitStruct.hEvent then
  SetEvent(SELF.WaitStruct.hevent)
  closehandle(SELF.ThreadHandle)
  closehandle(SELF.waitstruct.hevent)
  clear(SELF.Threadhandle)
  clear(SELF.WaitStruct.hEvent)
end
!Close the notification process
If SELF.WaitStruct.NotifHandle then
  FindCloseChangeNotification(|
  SELF.WaitStruct.NotifHandle)
  clear(SELF.WaitStruct.Notifhandle)
end
```

The code `SetEvent(SELF.WaitStruct.hevent)` sets the event object to signaled and effectively ends the API thread. Note also that for almost all handles, such as the thread handle and event handle, `closehandle()` should be called when you are done with them to free resources and avoid a small memory leak. The notification handle is closed by the special function `FindCloseChangeNotification`. If you didn't use the extra event object, when `FindCloseChangeNotification` is called the `NotifHandle` becomes signaled and the thread would end, but the code signaling the file change would be executed possibly causing some confusion. Having a separate event object to signal the end of the program is a bit cleaner and fairly simple.

As you can see from the example above, the event object was created on one (Clarion) thread and used on another (API) thread. Because of that, this object can be used to synchronize actions on two or more threads or processes. As a result, this type of object is frequently called a synchronization object. While a complete description of synchronization objects is beyond the scope of this article, I'd like to mention the one that's my favorite: it's called `CriticalSection`. Unlike an event object it is not a waitable object (i.e. can't be used in `WaitForSingleObject`) but is still very useful while at the same time being the fastest and lightest weight of all the synchronization objects. It can act like a traffic cop and keep more than one thread from accessing global variables at once. It's also simple to use. Just put the group in Figure 6 into your applications data section.

Figure 6. The CriticalSection group structure.

```
!API structure so not more than one thread
! accesses the globals.
CRITICAL_SECTION      GROUP
DebugInfo              ULONG
LockCount              LONG
RecursionCount        LONG
LockSemaphore         Unsigned
Reserved              ULONG
                      END
```


Don't give the fields of the group a second (or first) thought – you are not going to use them. At the start of the program before using the critical section add this to your code:

```
InitializeCriticalSection(address(critical_section))
```

```
!your code
```

```
DeleteCriticalSection(address(critical_section))
```

BadThreads Revisited

Think of this as an init and kill method. Recall at the start of [part one](#) I mentioned a program called BadThreads where two API threads (Thread10 and Thread20) each accessed global variables and interfered with each other. The three global variables were A, B, and C. To prevent this interference, you wrap the access to the global variables in a critical section, as in Figure 7.

Figure 7. Fixing BadThreads with a critical section.

```
Thread10 procedure(long param) !API thread
code
Loop
  EnterCriticalSection(address(critical_section)).
  A=10
  B=20
  C=A+B
  if C<>30 and ~doneflag then doneflag=10.
  LeaveCriticalSection(address(critical_section)).
  If doneflag then break.
end
return 1
Thread20 procedure(long param) !API thread
code
Loop
  EnterCriticalSection(address(critical_section)).
  A=20
  B=30
  C=A+B
  if C<>50 and ~doneflag then doneflag=20.
  leaveCriticalSection(address(critical_section)).
  if doneflag then break.
end
return 0
```

Notice that all usage of A, B, and C is between a pair of calls: EnterCriticalSection and LeaveCriticalSection. The code between the two calls is the critical section you are protecting.

Let's assume Thread10 arrives at EnterCriticalSection before Thread20. It gets inside the critical section. If Thread20 gets to EnterCriticalSection before Thread10 leaves the critical section, Thread20 is stopped. As soon as Thread10 finishes with the critical section, and executes LeaveCriticalSection, then Thread20 can enter the critical section. As a result of the traffic cop action, only

Thread10 or Thread20, but not both, can access the global variables A, B, or C at one time. Without the critical sections the two threads interfere with each other and produce bad arithmetic results within a second or two. Try out the BadThread sample code to see for yourself.

When all accesses to global variables are properly protected with critical sections, threads can be made to cooperate and peacefully co-exist. Fortunately for Clarion programmers, you rarely have to worry about this because the Clarion runtime normally ensures only one thread is active at a time. But if you get fancy and create a thread Clarion does not know about, then you have to protect any and all data you access which is declared outside the created thread.

The sample application in the downloadable called ChangeNotif.prj, uses the `cnotifcl` class I have been discussing to monitor the directory `c:\mydownloads` for any changes. All it takes is just one call to the `cnotifcl` `init` method. A user defined event, `EVENT:FileChanged` is posted when ever a file in the target directory is changed in any way.

Summary

Whenever you need to either wait for something to happen or complete a long process without making a Clarion application unresponsive, consider creating an API thread, and move the wait or process into it. Just don't call the Clarion runtime from the API thread since it isn't thread safe, and if you must access any global variables or resources from one or more API threads, consider using critical sections to avoid two threads accessing the same global variable or resource at once and interfering with each other.

[Jim Kane](#) was not born anywhere near a log cabin. In fact he was born in New York City. After attending college at New York University, he went on to dental school at Harvard University. Troubled by vast numbers of unpaid bills, he accepted a U.S. Air Force Scholarship for dental school, and after graduating served in the US Air Force. He is now retired from the Air Force and writing software for [ProDoc Inc.](#), developer of legal document automation systems. In his spare time, he runs a computer consulting service, Productive Software Solutions. He is married to the former Jane Callahan of Cando, North Dakota. Jim and Jane have two children, Thomas and Amy.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

[Main Page](#)[COL Archive](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)

Clarion News

July 25, 2000

[SoftVelocity Looking For XML Feedback](#)

SoftVelocity is looking for your thoughts on XML support. This and more in the SoftVelocity July 19 newsletter.

[Dalby Source Printer 5 Now Freeware](#)

The Dalby Source Printer version 5 is now available as freeware. Now with a 32 bit (Linder SetupBuilder 3) install program. You can also use DSP 5 to print text files, Java and C++ source, etc.

[Gitano G-RegPlus Suite Sale](#)

Gitano's G-RegPlus Suite is now on sale at 20% off. This applies to new sales, upgrades, and competitive upgrades. Customizations of G-RegPlus are available to meet your specific needs.

[RemFlash Beta 2](#)

RemFlash is really two templates in one - it will enable your apps with full reminder facilities, plus it will give a form of "Instant Messenger" facility when used on a network. No runtime fees, and compatible with all versions from CW2x through C5.5, ABC & Legacy. Demo available. The cost is \$99 for the duration of the beta program, thereafter \$149.

[CCS SQL for C5.5/ABC](#)

The CCS SQL templates for C5.5/ABC are available now directly from DeveloperPLUS. An upgrade for current C5/ABC version owners is also available. Product install available for immediate online fulfillment.

[XLIB Discount Expires July 27](#)

Alexey Solovjev's XLIB is on sale for \$45, or \$200 with source, through July 27, 2000. With XLIB you can create file, queue, and group structures at run time, create aliases and modify file structures on the fly, and more.

[Recover Owner ID For Password Protected .DAT Files](#)

Bobcat Systems can recover the Owner ID and password for protected Clarion (DAT) files from the first few dozen bytes of the file. Your confidential data stays in your hands.

[G-News Update](#)

A new issue of G-News is now available. This month's Tip & Trick: Quicken® style icons.

[Insight Beta 1c Released](#)

Beta 1c of CapeSoft Insight Graphing is now available. Support for queues is back in. One known bug - in clarion 5.5 the "local" mode compiling is not working, but that should be fixed shortly. If you get a GPF immediately on start-up then this is the likely problem. Insight Graphing is currently on special for \$199 - normal price is \$299. orders possible via

[Clarion 5.5 Gold Candidate: A First Look](#)
(Jul 25, 2000)[Legacy to ABC: There is Another Way!](#)
(Jul 25, 2000)[Using API Threads - Part 2](#)
(Jul 25, 2000)[July 2000 News](#)
(Jul 25, 2000)[Read The June 2000 News](#)

Sterling Data

Components
Freeware
Resources

www.sterlingdata.com

Developer PLUS

Your source for
development tools
and add-ons.

Your outlet for
application sales.

[Coming Soon: CWScript](#)

Steve Olensky has pre-announced a tailor-made ActiveX control and scripting language to allow Clarion users full access to industry standard ActiveX controls.

[Clarion Third Party Profile Exchange Update](#)

Download Product Scope 32 Bookmarks Viewer Version to view information on 157 products from 152 Clarion third party vendors. Features include quick access to web sites and order pages, demos, pricing, category, description and more.

[New LZip 2.50 Compression Library Evaluation Version](#)

A new LZip Compression Library 2.50 Evaluation Version for Clarion is now available. With LZip you can add industry-standard Pkzip and Winzip compatible data compression capabilities to your own Windows programs. LZip also supports high performance in-memory compression and decompression to give your users the benefit of minimal storage requirements. The LZip libraries have the latest generation zip compression engine built-in, and have no external dependencies such as MFC DLLs, runtime libraries or other resources. LZip 2.50 LZip Compression Library costs \$179.00 for a royalty-free usage license and comes with a free LSPack 2.0 license.

July 18, 2000

[New Bug Poster Template](#)

Novosys EDV GmbH has released the Bug Poster template, an extension which enables customers to enter bugs and enhancement requests which are emailed to the developer. Bug Poster ships with a small DLL and templates. Regular price is \$99, \$79 for the duration of the beta program. Trial version available.

[Buggy 1.03 Available](#)

An update to the Novosys Buggy bug reporting tool is now available to all registered users. The trial version is also being updated.

[Insight Beta 1b Released](#)

Beta 1b of CapeSoft's Insight graphing tool has now been released, and Beta 2 is expected shortly.

July 11, 2000

[IMPEX 4.0 Released](#)

IMPEX 4.0 is a major new update to Sterling Software's import/export product. New features include flat file import, wizard import interface, unattended and batch imports, user defined duplicate checking, like to SearchFlash, and more. Demo available.

[UltraTree Platinum Demo Update](#)

The UltraTree Platinum demos have been updated with new features including: tagging and untagging of single nodes, complete branches, or the entire tree; multiple footer rows, which are rows inserted into the list below a block of records, usually to hold a calculated value – the demo shows how to use these to do a standard invoice with subtotal, tax, discount, and total footers; headers and footers can be tagged, as well as data rows; and pressing the print button generates a report of all tagged rows.

[Insight Graphing 1.0 Beta 1](#)

Insight Graphing Beta 1 is now out. This template can read data directly from your data tables – no hand coding required. Graphs are supported on both windows and reports. This release supports bar, line, and pie graphs. Regular price is \$299, but during the beta Insight is available for \$199.

[NetTalk Beta Nearing Release](#)

Although not quite ready for a release all indications are that Beta 8 will ship in a few days. The major new addition is full SMTP and POP3 email support, including sending,

receiving and attachments. Also attachments can be automatically zipped before sending (and automatically encrypted using 168 bit DES encryption). At the other end your program can automatically decrypt, and automatically unzip the file ready for use. Regular price is \$299, but during the beta NetTalk is available for \$199.

July 5, 2000

[MySQL Goes Open Source, Attracts Investment](#)

The MySQL database server has been released under the GNU GPL. As an open source product, MySQL is now available free of charge for all supported platforms. The developers of MySQL have also formed strategic alliances with two companies. VA Linux will invest in MySQL development, will work on advanced support and service, and will host a MySQL project on [SourceForge.net](#). SourceForge now has over 5800 open source development projects. Progress Software is also investing and will offer the NuSphere MySQL distribution ([www.nusphere.com](#)).

[New Language Modules For SetupBuilder](#)

Linder Software's SetupBuilder 3.0 installation system now supports the following languages: Danish, English (US), French, German, Norsk (Bokmal), and Slovenian. Help is needed to translate the dialogs (14) and messages (~100) into other languages. If you ever find a free minute to translate the Language Module into, say, Portuguese, Spanish, Swedish, etc. please drop Friedrich Linder a line at friedrich@lindersoftware.com.

["One Touch" Date, Time, and Scheduling Tools Update](#)

An update of PD "One Touch" Date, Time and Scheduling Tools is now available. This has several fixes since the 6/26 release and two new features. An Appointment Browse Styles Tab simplifies creating cell level styles, and the old style Popup Calendar is now back by request. This does not have the week of the year list that is part of the new calendar and is somewhat larger. Includes some minor fixes and a demo appointment calendar.

[Add-On Writes To NT Event Log](#)

New from solidsoftware is CwEventLog. This add-on allows Clarion applications running under NT and Windows 2000 to write to the OS event log. Event logging provides a means to merge events from various sources into a single informative story.

[Clarion Mentioned In TDAN Article](#)

Mike Gorman has published a scholarly article titled "A New Paradigm for Successful Acquisition of Information Systems." The article proposes a nine step process to address the consistent failure of IT projects to come in on time and within budget, and points out Clarion's code generation strengths.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than [www.clarionmag.com](#), email covecomm@mbnet.mb.ca.