Clarion Magazine

This edition includes all articles, news items and blog posts from May 1 2011 to May 31 2011.

Clarion News

Read 20 Clarion news items.

Articles

Bob Foreman: An Open Letter to the Clarion Community

May 2 2011

No, Bob's not at SoftVelocity anymore, but he's still a Clarion guy and you can expect to see him around the newsgroups in the near future. NOTE: You do not need a subscription to read this article but, as this is a letter to the Clarion community and not the world at large, registration is required to read it. If you do not select any of the mailing lists when you register, you will not receive any email from Clarion Magazine. Your registration is confidential.

Tip of the Week: Ctrl-X Surprises

May 4 2011

The subject of using Ctrl-X to cut a line of text to the clipboard came up recently in the newsgroups, and Scott Ferrett pointed out an aspect of this behavior that some developers may not know about (I certainly didn't.)

Tip of the Week: Default Column Values

May 11 2011

Here's a C8 dictionary editor tip: You can set default values for your columns.

Tip of the Week: Handling Sheets and Tabs in C7

May 18 2011

The C7 designer handles sheets and tabs differently than C6. Here's what you need to know.

SQL Connection Strings, The Class

May 20 2011

It was eight years between Dr. Parker's first class and his second. It was eight days between his second and his third.

May Articles On The Way!

May 20 2011

I have a full slate of May articles in hand, but as I'm on the road this week publication of the first batch will be delayed until early next week. Some of these articles will be backdated to early May.
David Harms, Publisher

Clarion 8 Beta Notes for May 2011/Open Discussion

May 25 2011

Notes on the Clarion 8 beta for May, 2011, posted as comments to this article. Subscriber comments are also welcome.

Tip of the Week: Mass Control Changes

May 25 2011

As in C6, you can modify multiple controls at once in C7. But the process is a bit different.

Time And Again

May 27 2011

Time. Time and again it comes up on the news groups. Time, and its many calculations. Dr. Parker begins a new series of indeterminate duration.

Custom Code Folding

May 28 2011

Starting with Clarion 7, the source editor supports code folding. But not all statements are foldable. Russ Eggen shows how to add custom code folding using "magic" comments.

Time And Time Again

May 30 2011

Midnight rollover is one of the thornier problems in Clarion time calculations. Steve Parker looks at a commonly-used solution dating back to the LPM days: StarDates.

Source Code Library Updated

May 31 2011

The 2011 Q1 Source Code Library update is now available. The Q2 update is expected to be available at the end of June.

Clarion News

Clarion 8 Build 8274

A new release of Clarion 8 is available. This release fixes a variety of issues reported after the first C8 release.

Posted May 3 2011 (permanent link)

Fomin Report Builder 3.22

Fomin Report Builder version 3.22 is available. This is an immediate release to deliver Clarion 8 compatibility as well as some fixes and changes.

Posted May 6 2011 (permanent link)

IntelligentMail Barcode for C7.3(ABC)

The first release of the Intelligent Mail Barcode for C7.3(ABC) is available for current subscribers. Login to download the new install.

Posted May 6 2011 (permanent link)

Icetips Clarion 8 Builds

Arnor has made 11 product builds and 1 beta build available for download, all

Posted May 6 2011 (permanent link)

iQ-XML For Clarion 8

iQ-XML is now compiled and compatible with Clarion 8. Anyone doing a local compile (LIB) should be using this instead of a C7 version. In addition, the DLL is changed to IQXML80.DLL, and the templates have been updated to reflect all changes.

Posted May 6 2011 (permanent link)

SetupBuilder 7.4 Release Announcement

Lindersoft has released version 7.4 of SetupBuilder, which adds support for Clarion 8.

Posted May 6 2011 (permanent link)

Clarion 8-Aware Installer For LSZip

A Clarion8-aware install image for LSZip is available.

Posted May 6 2011 (permanent link)

Thin@ 1.5

Thin@ 1.5 fully supports ActiveX controls; the developers have been collaborating with Andy Wilton from Noyantis on making support for Noyantis Codejock Wrapper templates as well. Support for Noyantis ChartPro and ShortcutBar is ready. There's also other stuff, including drag and drop support, SVGraph & ASCII viewer support, improved thread engine etc.

Posted May 6 2011 (permanent link)

iQSync 1.42

iQSync 1.42 adds search and replace. In the Project Detail, select "Mass Change Folder..." and new option "Search and Replace". Now, all the To and From and Descriptions will change for you - such as "Clarion7" to "Clarion8".

Posted May 6 2011 (permanent link)

CapeSoft C8 Installs

All of CapeSoft's C7 installs are rebuilt and uploaded to include C8 compatibility. They have not tested everything yet, but will get the Clarion 8 specific issues sorted out one at a time as they arise.

Posted May 6 2011 (permanent link)

C8 Early Release Build 8312

A new C8 build is available to CSP participants.

Posted May 9 2011 (permanent link)

CHT And Clarion 8

If you visit the CHT forum and check out the newly posted article called: "CHT For C8 Installation: How To" you'll see a complete five step process explaining how to get your CHT templates up and running with C8. It includes three small downloads which you'll be asked to drop into various C8 \accessory\ areas. This is an incomplete install and will only work if you have a current subscription and you have Build 14A1.01 installed for Clarion 7, since you're going to be cross copying from the C7 \accessory\ area to the C8\accessory\ area.

Posted May 13 2011 (permanent link)

QuickBooks Connect 2.00 Alpha

The almost new from the bottom up V2.00 of Clarion QuickBooks Connect is in Alpha testing. This is not your father's QB Connect. There are only two classes (and one of those is an interesting string class). It can be hand coded (as in no templates required). It is extensible (as in you can extend the processing of QuickBooks modules beyond what is initially provided). It is fast (as in it took .38 seonds to put 10,787 customer names into a Queue after QuickBooks was finished creating the stream of query results). It is flexible (as in limit your QuickBooks query by any of the filters available in QB). There are currently no videos of the new system (been working too hard to get it ready) and it does not natively install on C8 yet but there is nothing in all of this that won't run on C8 as the DLL is not Clarion and the rest is source.

Posted May 13 2011 (permanent link)

SuperTemplates for Clarion 8

BoxSoft has released Super Templates (v7.40), compatible with Clarion 8. The passwords have not changed. For all purchase and upgrade issues (including passwords), please contact Mitten Software. Their e-mail address is answers@mittensoftware.com, and their phone numbers are 800-825-5461 and 952-745-4941.

Posted May 13 2011 (permanent link)

RPM for C8

RPM for C8 is available for current subscribers.

Posted May 13 2011 (permanent link)

Intelligent Mail Barcode for C8

Intelligent Mail Barcode installs for C6.3, C7.3 and C8.0 are available for current subscribers. The C8.0 install was updated 05-May-2011 to use CHM template help.

Posted May 13 2011 (permanent link)

Comsoft7 And C8

The bulk of Comsoft7's templates have installers to accommodate C8. The rest will follow as needed. Check the clarion versions supported column.

Posted May 13 2011 (permanent link)

G-Calc 7.01

Version 7.01 of the G-Calc template is now available. This release contains the following updates: C8 Compatibility; The Font name attribute can now be specified for the calculation result and tape list controls; BUG FIX: Global properties were being overwritten with default values when being used within a Multi-Dll solution. The update is free to all users who have an Active Maintenance Plan in place, or costs $29.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of the web site.

Posted May 13 2011 (permanent link)

TaskPanel 2.07

Version 2.07 of the TaskPanel wrapper template is now available. This is an Interim release. It contains the following updates: Codejock 13.4.2 -> 15.0.2 compatibility added; C8 Compatibility. The update is free to all users who have an Active Maintenance Plan in place, or costs $47.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of the web site.

Posted May 13 2011 (permanent link)

DockingPane 1.11

Version 1.11 of the DockingPane wrapper template is now available. This is an Interim release. It contains the following updates: Codejock 13.4.2 -> 15.0.2 compatibility added; C8 Compatibility. The update is free to all users who have an Active Maintenance Plan in place, or costs $47.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of the web site.

Posted May 13 2011 (permanent link)

CLARION MAGAZINE
READ. LEARN. SOLVE.

Home | Subscribe | E-Books | News | Blog | Store | My ClarionMag | My Lists | Contact

The ClarionMag Blog

Bob Foreman: An Open Letter to the Clarion Community

Posted May 2 2011

> Editor's note: We've received a number of emails from readers asking about Bob Foreman's status, whether he was still employed at SoftVelocity or had moved on to another company. Bob has been a prominent member of the Clarion community for many years; you can read a little more about him in his Icetips bio, published in 2002.
>
> Bob has also received a number of emails asking for clarification of his status. Here is his public reply, as provided to Clarion Magazine.

An open letter to all Clarion Developers

As we all grow a little bit older each day, we begin to realize how short this gift of life really is, and how each new day is truly a blessing. It really doesn't feel like 21 years has passed since the first time that I walked into the Clarion Software home office. It's amazing to think that my first experience with Clarion was on a Tandy 386 with a whopping 20 MB hard drive.

We have all come a very long way since then. During my time with Clarion Software/TopSpeed/SoftVelocity my favorite moments were spent in the classroom and at the developer conferences. It was (and still is) a great joy to meet users who have made a great living with this great development tool. I think that when you are on the front lines of support and deal with bugs and problem tracking on a daily basis you can sometimes lose track of the fact that many people have written some amazing programs with Clarion that have made an impact on this planet in nearly every field of business. Others who had started with Clarion have used that development philosophy in other ventures and at the same time continue to raise the bar. This all could not have happened without Bruce Barrington and Robert Zaunere's dedication, zeal and vision of Clarion that has impacted so many lives in a very positive way. My own work ethic and efforts are in part due to my attempts to aspire to their energies!

So it is that my own 21 years of Clarion have opened a new opportunity for me. In part this opportunity certainly embodies the Clarion spirit and will actually focus on what I love to do more than anything else - training and teaching. For these two reasons it was an offer and opportunity that I simply could not refuse.

So officially I am leaving the Clarion team for now, but unofficially Clarion will never leave me. I still plan to check in often on the newsgroups and will continue to offer help and suggestions when I can. I also have several projects written in Clarion that I will never abandon However, don't be surprised if I am not active in the immediate future, as my full attention will need to be focused on my new position and getting up to speed with the new technologies that it will be my responsibility to teach.

But I will be back! With Clarion 8 in beta and the Clarion.NET Application Generator nearly complete, 2011 will be an exciting year for Clarion users and SoftVelocity, and I look forward to watching the next great wave of development tools that are coming.

To this day, if I need a great Windows application, I still cannot think of a better tool to use than Clarion.

There are too many people to thank individually here; all of Clarion users and team members past and present that I have had the pleasure to work with makes my head spin. You are all a very passionate bunch, and what we have all done here in the last 21 years is something very great. You have all chosen your development tool wisely, and I think you know that.

All the best to all of you in all of your projects and passions. In this great information age we will certainly see each other again, if not in person, certainly in the cloud.

Sincerely,

Bob Foreman

Article comments

by Dave Harms on May 2 2011 (comment link)

Bob, thanks for the udpate! You've been a great advocate for Clarion over the years (sounds like that's not changing<g>) and I appreciate all you've done as a member of the community and in your role at SoftVelocity and its predecessors.

I wish you all the best in your new job, and I look forward to seeing you back in the newsgroups!

by Peter Hermansen on May 3 2011 (comment link)

Hi Bob, It's a great loss for the community to see you gone as you've been of tremendous help to all of us here, almost "forever". However, I understand that sometimes it's time for a change - for whatever reason - and as you say, we're not getting younger. So, obviously now the time was right for you to make such change and I wish you all the best in your new life and I do hope that we'll stay in touch in the news groups or elsewhere.

Peter Hermansen

by slimboywynn on May 3 2011 (comment link)

Bob, many thanks for everything you have brought to the clarion community over the years. I look forward to still seeing you around and involved with everything 'clarion'.

Good luck.

Colin.

by Bruce Johnson on May 3 2011 (comment link)

Bob,

It has always been a pleasure and a privilege to work with you over the years. It seems we were always running into each other in far-flung corners of the world, and I know your impact on

Clarion programmers of all generations has been immense.

All of us here at CapeSoft wish you success with your new endeavors, and we hope to see you again sometime.

Don't forget to keep on bowling - those pins won't knock themselves down.

Cheers Bruce

---

by JP (DMC) on May 3 2011 (comment link)

Bob,

We missed you. We will miss you. I miss your "normalization" role in commiunications between SV and us. I'll miss your singing if I go to next Denver CLDC. I hope someday "something" will allow you to compensate the part you do not mention and to give you back the part you prefer and allow you to "focus on what I love to do more than anything else". Take care Bob. JP (the Best bad Boy) .

---

by Graham Dawson on May 3 2011 (comment link)

Hi Bob,

Many, many thanks for all your help in the usergroups, you've always been a gentleman.

You're absolutely right to grab those opportunities, life really is too short to put things off.

All the best for the future.

Graham Dawson

---

by Stu Andrews on May 3 2011 (comment link)

Bob,

Was a real privilege to meet you at the Aussie Devcons. Your passion encouraged me, and I'm a passionate bloke!

The Almighty grant you a fulfilling life onwards and upwards.

Stu

---

by AndrÃ© LabuschagnÃ© on May 3 2011 (comment link)

Bob

It was a real privilege to meet you at the Oz Devcons and see the passion you have for this development tool. It really inspires one. Your contribution has been immense.

Change is the stuff that life is made of and I wish you success with the opportunity that has presented itself to you. I look forward to your contributions on the forums and, who knows, we may even see you back at SV some time in the future.

All the best to you and your family.

Cheers Andre

---

by Brian Reid on May 3 2011 (comment link)

Bob,

Congratulations on your new adventure. I remember the first class I took at Clarion back when CW 1.0 was being released. You did a tremendous job for Clarion then and have ever since. Known to many of us as the heart and soul of the clarion organization, you will certainly be missed. Good luck to you and your family. May the sun shine on all your days ahead. Cheers, Brian K. Reid

---

by Leo Lakota on May 3 2011 (comment link)

Bob,

THANKS and all the best for the future !!

---

by Mike Hanson on May 3 2011 (comment link)

Thanks for all your help and guidance over the years, and have a great time with your new path in life. All the best!!! :)

Mike Hanson

---

by Geoff Bomford on May 3 2011 (comment link)

Bob,

For any Clarion user who has ever attended a devcon, or training session, you were the face of Clarion and your friendly personality, honesty, communication prowess and technical know-how will be sorely missed. I wish you good health and the best of luck in whatever endeavors you undertake in the future!

Regards, Geoff Bomford, Sydney Oz.

---

by Bill Quesse on May 3 2011 (comment link)

Bob,

I hope you have great success in your new work. Thanks for everything you have done for all of us.

Bill Quesse

---

by Russell Eggen on May 3 2011 (comment link)

Bob,

You are the one that inspired me to do better in teaching Clarion, presentations with lots of meaningful content. I always considered you competition in a friendly way. It was fun trying to come up with lesson plans and presentations that you would say "Wow!"

I can't thank you enough for that as that made me better as well. But you will never teach me bowling! LOL

Don't stop having fun!

---

by Geoff Spillane on May 3 2011 (comment link)

Bob,

Thank you for being you. A passionate man in everything you do. I feel very privileged to have met you on several occasions. I hope you know you've messed up my big plans for next year's Aussie DevCon . All the best and keep well. Keep up the bowling.

Best Regards, Geoff

---

by Axel Fenger on May 3 2011 (comment link)

Bob,

thanks for your great contributions to Clarion and its community. All the best for the future, Axel F.

---

by Joe Tailleur on May 3 2011 (comment link)

Bob,

Thanks for your wonderful support every day in the news groups, in email, and in the training sessions at Devcon. We will miss your kind hearted spirit and your invaluable knowledge so happily shared with anyone in need of a hand.

Joe Tailleur.

---

by David Jung on May 3 2011 (comment link)

Thank you!

--David

---

by Ned Reiter on May 3 2011 (comment link)

Bob,

I always benefited from your sessions at numerous Devcons and enjoyed your company on more than one occasion. I wish you the best in your new endeavor and look forward to your continued participation in the Clarion community.

---

by Arnor Baldvinsson on May 3 2011 (comment link)

Hi Bob,

Thanks for all the help in the past 15 years or so! You will be missed! I wish you all my best in your new endeavors:)

Best retgards,

---

by Robert N. Bass on May 3 2011 (comment link)

Bob, Thanks. Good luck. Bobby in Charlotte

---

by Jane Fleming on May 3 2011 (comment link)

Bob,

Thanks for all you've done to bring clarity.

Forks in roads are challenges as well as opportunities, and I hope yours is invigorating and rewarding.

All best wishes,

Jane Fleming

---

by Skip Williams on May 3 2011 (comment link)

Bob,

Thanks for 21 years! You are a true gentleman and professional. Look forward to your continued involvement on the newsgroups.

Good luck and Godspeed!

Skip

---

by Tom Hebenstreit on May 3 2011 (comment link)

Sure gonna miss you, and I hope you DO pop up at DevCons -- you've always been a pleasure to talk with, and a great teacher besides!

Tom H

---

by Sean Hennessy on May 3 2011 (comment link)

I echo the sentiments of all the others.

live long and prosper !

Sean H in Aus.

---

by Rocky on May 3 2011 (comment link)

Bob,

I has been a pleasure to know you. Your dedication to the Clarion product and the Clarion community have been outstanding. Thanks for the years!

Rocky

---

by Geoff Robinson on May 3 2011 (comment link)

Bob

it was great to meet you at the last couple of Oz DevCons in Eden and attend your training sessions. Your passion, helpfulness and gentlemanly manner were always present, whether on-line or in person. Thanks and good luck with your new path!

Geoff R

Melbourne, Australia

---

by Roy Brubaker on May 4 2011 (comment link)

Bob, From the conversations with me (and my ex-business partner) in 1991 to discussing the pros and cons of your new opportunity, you have always been the penultimate professional Clarion source and just plain excellent human being.

Best of luck in your new situation and best to your family who are awesome in their own right.

Roy Brubaker

---

by Merle Windler on May 4 2011 (comment link)

Bob, Thank you Thank you Thank you for all the help you have provided to all of us over the last two decades.

It's been fun seemingly chasing you around America, traveling to new places to attend your Clarion training classes. It was you that eased my fears when the new CFD3 seemed unsurmountable...Guiding my transition into the 'world of Windows', and training on transitions from CFD2 to CFD3, CW thru C7.

It has been a great joy to know and work with you over these many years. It's comforting to know that your knowledge, integrity and spirit is not leaving the Clarion community... and you will continue to be a Clarion user, fan, and supporter.

Hope to meet up with you again at Clarion gatherings or maybe a Tommy Emmanuel or Chet Atkins Appreciation Society guitar players convention.

Best wishes My Friend.

Merle Windler

---

by Robert Hutchison on May 4 2011 (comment link)

Bob

Thanks for the help you have given me over the years. All the best for the future.

Robert Hutchison

---

by Paul Konyk on May 4 2011 (comment link)

Bob,

I can only thank you for your efforts and wish you all of the best in your new endeavor.

Your enthusiasm and joy of spreading the Clarion brand will be missed.

Paul Konyk

---

by Arturo Rivera on May 5 2011 (comment link)

Gracias por la pasion de tú trabajo en Clarion!

---

by Tony York on May 12 2011 (comment link)

Why am I always the last to find out these things. I know why, it's because I was busy programming with Clarion and I didn't notice the time go past.

Bob, it was great to meet you at the Aussie Devcons, and I wish you well in your future ventures.

Tony York

---

by Philip Prohm on May 13 2011 (comment link)

Bob, you're an inspiration. Thank you and best wishes in your new endeavours.

Philip Prohm
NSW

---

by Ali Ben MaÃ¢tallah on May 14 2011 (comment link)

Bob, Thank you. I wish you all the best in your new job. Ali from Tunisia.

---

by Michael Gorman on May 14 2011 (comment link)

Bob:

Just went through all the comments above and I cannot think of anything more that could possibly be said, and for ME, that's a real earth shaking event....

So, I echo all those from above and know that you'll be working with others of like mind.

I look forward to keeping in touch...

Regards,
Mike Gorman

---

by Stephen Mull on May 18 2011 (comment link)

Bob,

Just read this... wow, what a surprise. I have been working with both you and Clarion for over 20 years now, that is a long time for anything IT related. While true that most good things come to an end, they are often followed by something even better, just something different of course. I sincerely wish your move to be a 'something even better' event for you. I want to thank you for all of the assistance over the years, I wish you all the best! --Steve

BACK TO TOP

Tip of the Week: Ctrl-X Surprises

By Dave Harms

Posted May 4 2011

The subject of using Ctrl-X to cut a line of text to the clipboard came up recently in the newsgroups, and Scott Ferrett pointed out an aspect of this behavior that some developers may not know about (I certainly didn't.)

When you use Ctrl-X to cut some selected text, everything works as you'd expect. You press Ctrl-V to paste and the text is inserted at the current cursor.

But if you use Ctrl-X without selecting one or more characters, something quite different happens.

First, the entire line in which the cursor sits is deleted.

If you select some text and press Ctrl-V, the line is pasted as a replacement for the selected text.

But if you do not select any text, if your cursor is just sitting minding its own business in a line somewhere and you press Ctrl-V, the text is *not* pasted at the current location; instead the line of text is inserted above the current line.

This behavior is not unique to Clarion/SharpDevelop - it's how Ctrl-X works in the Visual Studio editor also.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors (ASJA).

Article comments

by Russell Eggen on May 12 2011 (comment link)

Nice tip! I did not know this also.

Russ

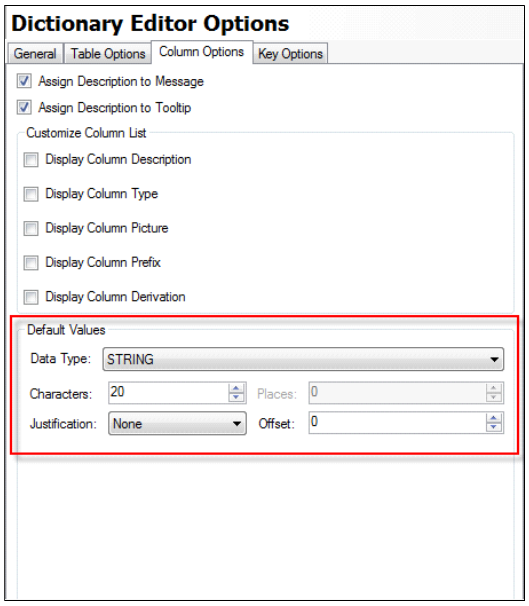
BACK TO TOP

Tip of the Week: Default Column Values

By Dave Harms

Posted May 11 2011

Clarion 8 adds a number of nice features, but one that's grabbed a lot of attention is a new Dictionary Editor option for default column values (Figure 1).



[✓] Assign Description to Message
[✓] Assign Description to Tooltip

Customize Column List
[ ] Display Column Description
[ ] Display Column Type
[ ] Display Column Picture
[ ] Display Column Prefix
[ ] Display Column Derivation

Default Values
Data Type: STRING
Characters: 20
Places: 0
Justification: None
Offset: 0

Figure 1. Default column values

Obviously not all your strings are going to be 20 characters, but you have options here for all of the simple data types. For instance, it's quite likely that if you use decimals you have a preferred decimal format. Where appropriate you can also set a justification and an offset.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors (ASJA).

Article comments

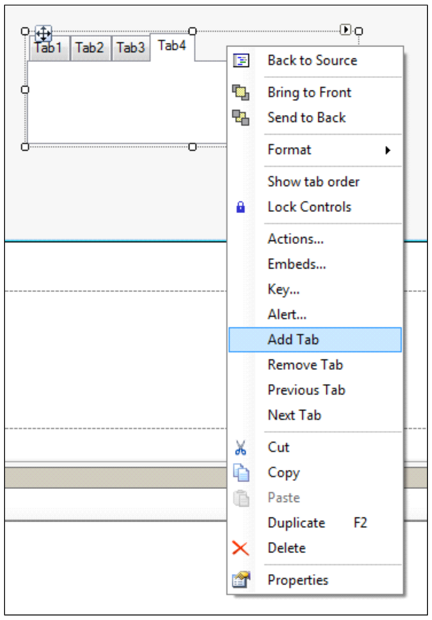
BACK TO TOP

Tip of the Week: Handling Sheets and Tabs in C7

By Dave Harms

Posted May 18 2011

There are a number of different ways to work with SHEET and TAB controls in Clarion 7, particularly when it comes to adding/removing tabs. You can right-click and choose to add or remove a tab via the context menu (Figure 1), or you can use the Tasks menu (Figure 2).



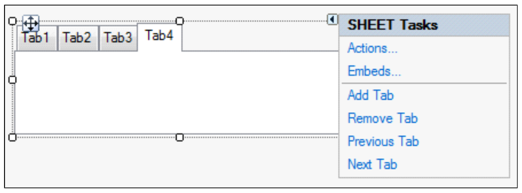
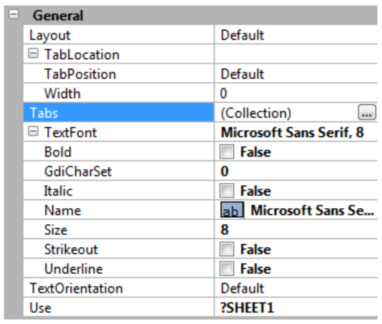


Figure 2. The Tasks menu

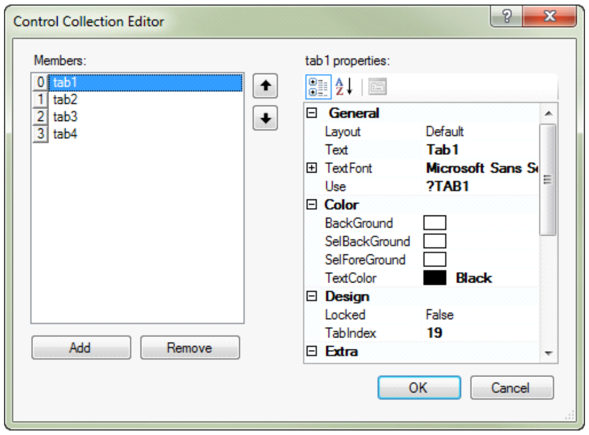
But the most complete access to sheet and tab settings is via the Tab Editor. To bring up this editor, select the sheet and look at the Properties pad. You'll see a Tabs entry with a (Collection) value. Click in that field and an ellipsis button will appear, as shown in Figure 3.



General	
Layout	Default
TabLocation	
TabPosition	Default
Width	0
Tabs	(Collection)
TextFont	**Microsoft Sans Serif, 8**
Bold	☐ **False**
GdiCharSet	**0**
Italic	☐ **False**
Name	**Microsoft Sans Se...**
Size	**8**
Strikeout	☐ **False**
Underline	☐ **False**
TextOrientation	Default
Use	**?SHEET1**

Figure 3. Sheet properties

Click on the button and you'll see the tab editor as shown in Figure 4.



General	
Layout	Default
Text	**Tab1**
TextFont	**Microsoft Sans S**
Use	**?TAB1**
Color	
BackGround	☐
SelBackGround	☐
SelForeGround	☐
TextColor	■ **Black**
Design	
Locked	False
TabIndex	**19**
Extra	

Figure 4. The tab editor

As an aside, the tab editor is, as far as I know, simply an implementation of the .NET `CollectionEditor` control. (Similarly, the property pad is an implementation of the .NET `PropertyGrid` control.)

The context menus (Figures 1 and 2) give you ready access to basic sheet/tab management, but for full control you'll want to use the tab editor.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors (ASJA).

Article comments

by Lee White on June 1 2011 (comment link)

And, for us keyboard junkies, where Dave uses "right-click" in the first paragraph you can also use the context menu key on your, wait for it, keyboard! That's the handy key between your right "Windows" key and your right Ctrl key.

I'm just say'n!<g>

BACK TO TOP

CLARION MAGAZINE
READ. LEARN. SOLVE.

Home | Subscribe | E-Books | News | Blog | Store | My ClarionMag | My Lists | Contact

SQL Connection Strings, The Class

By Steven Parker

Posted May 20 2011

It was eight years between my first class and my second. It was eight days between my second and my third. I immediately thought of the TV show “Criminal Minds” because the term “devolving” immediately came to mind. Was I in danger of becoming a serial class writer? Was I craving an adrenaline rush, craving a classes fix?

The danger, now that I’ve realized the place of classes in my tool belt, is real (and, no, I haven’t yet had DAB’s “eureka moment”). I may be a serial object maker....

My first class was the result of laziness (“A Class for Tagging”). My second was a product of necessity (“A String Class”). In doing that second class, I figured out that classes were appropriate when:

1. the procedure or function needs to be called from multiple places
2. the procedure would be an appropriate addition to a toolbox DLL (a.k.a. what used to be called a “personal function library”)

or, in lieu of #1:

3. the procedure will / could be called from multiple apps.

My second and third bullet points, on their face, appear redundant. After all, what’s the point of a DLL if the procedure is not intended for use in multiple apps? However, “appropriate in a DLL” is more a way of thinking about a procedure. For me, psychologically, the question is “If I had a toolbox DLL, is this something I would put in it?”

If the procedure needs to be called from multiple places but only within a single app, a local procedure seems more logical to me. Creating a procedure within an .APP is significantly less work than creating a class. The price is that it cannot be used elsewhere. Flexibility vs. ease of coding / maintenance. Dilemma….

> My dear editor claims that it’s a question of experience, that if you’re familiar with writing classes, they are no more difficult to write than a procedure, and far more flexible. We both claim that the other retains the right to be wrong (and is wrong).

My current class is not going to be called from multiple places in any single app (violating proviso #1). It would, however, be appropriate in a toolbox and it will be called from multiple apps. Two out of three, it makes the cut.

Here’s the problem I need to solve this time. I’m moving an app, actually the first of several, from TPS and Btrieve to MS-SQL. The very first thing a Clarion app must do to talk to SQL is construct the connection string.

In the past, SQL projects I've worked on kept the connection string (or the pieces from which the string could be built) in an INI file or in a local TPS file. Storing the required information in an INI or TPS file means

- each instance of the EXE must have a copy of the file, or access to the file in a shared location
- the first time an instance of the app starts up, the user has to create the file's contents, or the file has to be supplied with the install (not an option in many cases and not for me,)

In my current project, having the user do anything at all to create the information is unacceptable. Luckily, the installer handles checking for MS-SQL Express, installing it if necessary. It also runs a db creation script, if necessary.

The installer, obviously, knows the SQL instance. The db name is fixed too (I'll be using `SouthBreeze` for this article). With the `TRUSTED_CONNECTION` switch, it should be unnecessary to ask the user for anything. The only variable in this picture is the SQL instance name, i.e., the SQL server. The installer writes an XML file containing this information to a fixed location.

That location happens to be C:\Program Files\< companyName >. For the purpose of this article, I'll use C:\Program Files\CMag. The file's name is fixed and, for this article, assume the file name is shpDB.xml.

Given that the db name is fixed and, because I'll be using `TRUSTED_CONNECTION`, all I need in the typical connection string construct:

```
LOC:ConnectionString = Clip(SQLServer) & '; ' & |
  Clip(SQLDataBase) &    '; TRUSTED_CONNECTION=Yes;'
```

SQLServer is a variable containing the server instance name, and is stored in the shpDB.XML file, for example:

```
<?xml version="1.0" encoding="utf-8" ?>
   <DataBaseConnection>
     <ServerInstance>.\sqlexpress</ServerInstance>
   </DataBaseConnection>
```

So, what I need to do is find, open and read C:\Program Files\shpDB.xml and get the information between the `<ServerInstance>` tags.

Simple, right? If I was writing this as a procedure inside an app, it would be fairly simple. Here's the prototype:

```
GetConnectionString Procedure(),String
```

Which is called as:

```
GLO:ConnectionString = GetConnectionString()
```

In Global Data:

```
 GLO:INIFileName String(256)
```

In Local Data:

```
LOC:ConnectionString String(256)
SQLDatabase          STRING('SouthBreeze')
```

```
XMLFile  File,Driver('ASCII'),Name(GLO:INIFileName),Pre(XML)
Record     RECORD
DataLine     String(1024)
           End
         END

Pos1    Long
Pos2    Long
```

The `GetConnectionString()` code is straightforward:

```
Clear(LOC:ConnectionString)
GLO:INIFileName = 'C:\Program Files\CMag\shpDB.XML'
If ~EXISTS(GLO:INIFileName)        ! shpDB.xml not found
    MESSAGE('Cannot find shpDB.xml, containing SQL connection information.'&|
   '||Application will be terminated.', |
   'Error', ICON:Hand)
   Return LOC:ConnectionString
 End

Open(XMLFile)
If ErrorCode()       ! can't open shpDB.xml (for reason other than non-existance)
    MESSAGE('Cannot open shpDB.xml, containing SQL connection information.'&|
   '|Errorcode ' & ErrorCode() & ': ' & Error() &|
   '||Application will be terminated.', |
   'Error', ICON:Hand)
   Return LOC:ConnectionString
 End

 Set(XMLFile)
 LOOP
   Next(XMLFile)
   If ErrorCode()
     Break
   End

   If Instring('SERVERINSTANCE',Upper(XML:DataLine),1,1)
    Pos1 = Instring('>',XML:DataLine,1,1)
    Pos2 = Instring('<',XML:DataLine,1,Pos1)
    SQLServer = XML:DataLine[Pos1+1 : Pos2-1]
    Break
   End
 End
```

`Note:` I am not using an XML parser to read the XML file. Because the file is quite small and because I am interested in only one line, I feel that the Loop and Instring checks are sufficient for the purpose.

```
If ~SQLServer    ! couldn't read ServerName
   MESSAGE('Could not read Server Instance from shpDB.xml.'&|
   '||Application will be terminated.', |
   'Error', ICON:Hand)
   Return LOC:ConnectionString
 End

 LOC:ConnectionString = Clip(SQLServer) & '; ' |
  & Clip(SQLDataBase) & '; TRUSTED_CONNECTION=Yes;'

Return LOC:ConnectionString
```

I have actually written and tested this procedure. It works just as expected.

Now, how do I put it into the hands of a dozen or so developers on our team who may need it? Did I mention that some of the apps that will need this are Clarion and some are ABC?

I could create a black box DLL (Bjarne Havnen's "Generic DLLs The Template Way" shows how to make creating black box DLLs easy). I can't really distribute TXAs because of the mixed environments (as my esteemed editor notes: if you really want to blow up a Legacy app, try importing an ABC TXA). Or I can make a class.

SQLFunctions Class

The INC file is totally straightforward:

```
OMIT('_EndOfInclude_',_SClass_)
_SClass_ EQUATE(1)

SQLFunctions   CLASS,TYPE,MODULE('SQLFunctions.clw'),LINK('SQLFunctions.clw')

Construct         Procedure
Destruct          Procedure
GetConnectString  Procedure(), String
GetConnectString  Procedure(*String pConnectStr),Byte
GetServerName     Procedure(String pServerName),Byte
              End

_EndOfInclude_
```

The only unusual thing is that I did was to set up two ways to call the procedure. Hence, the `GetConnectionString` procedure is overloaded and the procedure label appears twice in the prototype list. One way, I call it as:

```
GLO:ConnectionString = GetConnectionString()
```

The other way, I pass `GLO:ConnectionString` to the procedure and get back the error:

```
If GetConnectionString(GLO:ConnectionString) <> Level:Benign
  Post(Event:CloseWindow)
End
```

Both methods of getting the information are shown in the test app which accompanies this article.

> Speaking of the test app: create \Program Files\CMag and copy the included shpDB.XML file into it. Run the program and it will return the connection string. Change the `<ServerInstance>` in the file and run the program. It will return the new connection string. Delete the file and run the program. It will return an error. (I `STOP` to show you the connection string, error or not.)

The code, above, should move into a class' CLW file virtually unchanged. In fact, the only issue would appear to be the need for an ASCII file within the class.

Somewhere, I remember hearing that compound data structures have to be declared outside a class. But there is nothing in the documentation barring declaring a file in a class (or within any method in a class). So, I tried it. The header of my class CLW looks like this:

```
    MEMBER

     Include('SQLFunctions.inc'),ONCE



    Map
    End

  LOCINIFileName      String(256),Static
  XMLFile  File,Driver('ASCII'),Name(LOCINIFileName),Pre(XML)
  Record     RECORD
  DataLine     String(1024)
             End
           END
```

The rest of the CLW is just the code shown above.

It works. The expected connection string is returned. So much for the “problem” of declaring files in classes. (As John Dunn pointed out, because the file declaration is not in the INC file, technically, it is not “in” the class, not a part of the class. Because it is in the CLW, before the method declarations, it is a resource that the methods are permitted to use.)

Extensibility

Soon after I was satisfied that the class was working, our install builder decided that C:\Program Files\<companyName> was an inappropriate location. The decision was that all future installs would write the XML file to C:\Users\Public\Documents\<companyName>.

This means that I have to check the preferred directory and, if shpDB.XML is not there, check the “old” folder.

Easy peasey.

```
  GLO:INIFileName = ' C:\Users\Public\Documents\CMag\shpDB.XML'
  If ~EXISTS(GLO:INIFileName)   ! not in public
    GLO:INIFIleName = 'C:\Program Files\CMag\shpDB.XML'
```

```
   If ~Exists(GLO:INIFileName) ! check old location
     MESSAGE('Cannot find shpDB.xml, containing SQL connection information.'&|
    '||Application will be terminated.', |
    'Error', ICON:Hand)
    Return LOC:ConnectionString
   End
 End
```

(The actual code checks both C:\Program Files\CMag\shpDB.XML and C:\Program Files (86)\CMag\shpDB.XML. I have not, however, included this part in the attached class files – it is easy enough to add.)

"Just" do a straight-forward nested search. If the file is found in public documents, I'm done. Otherwise, check Program File. If the file is in neither place, error out.

In the real class, however, I do not hard code C:\Users\Public\Documents\<companyName>. We have an in-house template/DLL that detects our "public folder" . A variant version of this class takes that function as a parameter and constructs the fully qualified DOS file spec at the top of the code.

A second set of eyes

The discussion above is almost entirely about the problem and how I solved it. Behind the scenes, however, lays an instructive story.

As mentioned, I first created the `GetConnectString` procedures inside an app. Because I am so used to working in app files, developing `GetConnectString` this way made debugging it easier (for me).

Once I was satisfied that it worked as desired, I cloned the code to the class files, made the necessary change to use the class methods and retested.

Then I decided to create a variant version of the procedure, as mentioned. I would not normally do this. I did so only because I was creating the methods for other developers with very different coding preferences. Some prefer passing external variables, some prefer getting error codes, some prefer.... Okay, it was also a bit of a learning opportunity for me too.

To create the overloading version, I did what any right-thinking Clarioneer would do, I copied, pasted and modified. Lazy is as lazy does.

Everything worked, I was thrilled. So, I wrapped it up and shipped it out. My eagle eyed editor, however, noticed a substantial amount of code duplicated (I have included the original INC and CLW files in the download available at the end of the article – pay particular attention to the two `GetConnectString` method declarations).

Three things happen in `GetConnectString`. First, I find the file. Second, I read the file. Lastly, I construct the connection string.

As Mr. Editor pointed out, the first two, with minor modificiations can be removed from both versions of `GetConnectString` and placed in a method, called from `GetConnectString`. (The method could, indeed, be `Private`, since it will not be called from anywhere but `GetConnectString`.)

Why "minor modifications?" Because one version is designed to return an empty string on an error and the other returns and error flag. Compare the final code in the CLW and you will see that having a second pair of eyes look over your code can cause great efficiencies in your finished code.

Summary

I worry. I seem to be getting the hang of this class stuff and I'm not sure I like that.

But I do know that the development cycle for this classes went faster than had I made a DLL. Using a class eliminated a generate/build cycle. I guess that's not bad.

I quess I am coming over to the dark side....

Download the source

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since version 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.

Article comments

BACK TO TOP

Clarion 8 Beta Notes for May 2011/Open Discussion

Posted May 25 2011

There's a lot happening on the Clarion 8 beta front, with a second release already posted. Rather than post a new article with each release, I'll add comments here and I encourage you to do the same.

This article is for May 2011 - look for a new beta comments article for each month of the beta.

Article comments

by ClarionMag user on May 5 2011 (comment link)

Bob Z has reported that a possible cause for disappearing and/or all-black menu items has been found, and is being tested.

The menu problem is intermittent, and has been reported as far back as Clarion 6.3.

by ClarionMag user on May 5 2011 (comment link)

Some users have encountered a very strange problem with certain entry fields in the AppGen, such as when typing a procedure name in the Actions dialog. Certain keystrokes (such as A E K S and R) are not permitted. Pierre reports that this has been fixed internally. Meanwhile the alternative is to type the text into Notepad or some other editor and paste into the entry field.

by ClarionMag user on May 5 2011 (comment link)

Is C8 ready for production? Robert Paresi, who has not been willing to release C7 apps into the wild, plans to go into production with a C8 version of his flagship app at 4400 sites on June 1. Stay tuned.

by Russell Eggen on May 8 2011 (comment link)

As of C8.8274, its not quite ready for solid production work, but I can use it for simple apps and I have deployed one of these into the wild. There are still some IDE issues that make it more difficult than C7, and while none that I've found are show stopping in nature, it does slow me down when compared to C7.

I expect the next build (which some of these issue are reported fixed) will be a lot closer. The current rumor for this new build is Monday May 9th (although I would not mind a delay to get a few more fixes).

C8 does bring a lot more to the table than C7, so its potential is exciting. I do very much approve of the new key definition layout in the dictionary. There are many new features in C8, that should definitely add to a developer's productivity. In the short time that C8 was released, we've already had two builds in total, which so far shows a faster turnaround time by Softvelocity.

I'm sure there will be more to report in the next build, hopefully that its smooth enough to increase production compared to C7. When that day arrives, C7 is retired here.

Russ Eggen

RADFusion International, LLC.

---

by Dave Harms on May 9 2011 (comment link)

A couple of reports of the IDE locking up and showing a "Parsing..." message. Joe Tailleur is noticing this happening after being in and out of several large solutions.

Not everyone agrees that C8 is faster - some find C8 slower than C7.

SuRF (Scott R Ferrett) is showing up in the newsgroups again, responding to C8 questions and reporting on bug fixes.

You should be able to use the C7.3 In-Memory driver with C8 with no problems.

There are a number of requests now to be able to specify an icon size along with an icon, where the icon file contains multiple sizes.

---

by Dave Harms on May 9 2011 (comment link)

New build out!

Release Notes: C8.0.8312

Fixes/Features/Changes

CHANGE: The default redirection file now uses the %Configuration% macro to locate obj, res, rsc, lib, map and FileList.xml files

CHANGE: The default redirection file now has an entry for finding *.chm files in %BIN%;%ROOT%\Accessory\bin

CHANGE: You no longer have to enter a label as the first thing you do when adding new item to the data pad or the dictionary editor

CHANGE: Moved Report/Window Preview button to the left of the toolbar so that its always available

CHANGE: Template Zoom dialog (F10 key) normalized icons and font size

CHANGE: Shorten long #prompt in brwext.tpw

FIX: You could define a MEMO or BLOB outside a file structure using the data pad

FIX: You could not define an &amp;ANY with the Data pad

FIX: Blurry icons in the Window/Report Designer

FIX: Conditions when item without text can be treated as a separator were incomplete

FIX: Error reported for the SKIP attribute on STRING controls in reports

FIX: If #EMBED is defined within #AT-block for another #EMBED with the HIDE attribute, user-defined responses for that embed displayed as orphaned in the Embed Tree

FIX: Merging of template changes could affect procedure's modification date/time

FIX: Multiple right mouse clicks and/or rapidly repeated access to popup menu's could fail/cause a crash

FIX: Origin of the image rectangle was calculated incorrectly if a button had the RIGHT attribute and its height is sufficiently less than image width

FIX: PRE attribute treated as incorrect on parsing declarations of LIKE variables from TXA

FIX: Popup menus were not always removed from the menu hash table

FIX: Probable incorrect deformatting of time or date value if component is equal to 0

FIX: Propagating some flags from the "last on line" field to its region

FIX: Sometimes when closing an App before it completely finished opening an exception could be generated

FIX: Sometimes some IDE toolbar buttons were hidden instead of shown in the overflow dropdown

FIX: The root node for orphaned embeds was not displayed in the Embed Tree

FIX: Unexpected window could be activated after closing an MDI child window

FIX: Incorrect values returned on evaluating some %Driver-dependent symbols

FIX: Variables of ASTRING type without initial value exported to TXA incorrectly

FIX: blurry icons in Embed Editor (prev/next)

IMPROVEMENT: Template driven dialogs can now have any number of pages (#TABs in top-level #SHEET)

IMPROVEMENT: The Help, Validity Checks and Controls tabs are not displayed if a field is a reference field

IMPROVEMENT: Listbox general tooltip is not required anymore to activate per-cell and per-column tooltips

PTSS 35051: Weird results for File Driver Symbols

PTSS 37912: Z order of MDI windows gets corrupted when closing MAX window

PTSS 37932: Extended Browse Options : LIST/DROP/COMBO - READONLY

PTSS 37940: deformat doesn't format data with @T4 picture well

PTSS 37956: SKIP Attribute not recognized

PTSS 37960: Modified Dates on all procedures changed to TODAY on import

PTSS 37961: Problem with column tooltips not showing

PTSS 37965: Export/Import Txa Problems

PTSS 37972: Export/Import of a Variable with type ASTRING

PTSS 37974: POPUP GPFs

PTSS 37975: Missing ICON on BUTTONs with RIGHT attribute

PTSS 37977: MENUs with RIGHT appear too far to the right

PTSS 37979: Setting PROPLIST:LastOnLine affects PROPLIST:RightBorder

REGRESSION: under some circumstances you could not enter a, e or k in controls in Template dialogs

---

by Dave Harms on May 10 2011 (comment link)

From Bob Z, here's a familiar way to post C8 bug reports:

> We've decided that for the time being it'll be simpler for all if we just continue to use the PTSS system found at: http://problemtracker.softvelocity.com/clarion7/ rather than starting a new PTSS/database (at this time); the Product dropdown field has a selection for Clarion8.

---

by Dave Harms on May 10 2011 (comment link)

Anthony Robinson reported a problem with the embed tree going missing, followed by an eventual crash. SV has reproduced the problem. And there are other reports of instability with browse embeds (they become orphaned), but I don't know if the two issues are related. There was some work done on the embed system for this latest build, and perhaps that has introduced a new issue.

---

by Richard Dafler on May 10 2011 (comment link)

Bob Z. mentioned "Edit data as text..." That's the first time I've heard that, and I have no idea what that means in practice. Could someone point me to some documentation or work up a quick example. Thanks!

---

by Dave Harms on May 10 2011 (comment link)

Rick, there's a new button menu at the top of the Data pad. Maybe that's what you're thinking of.

---

by Richard Dafler on May 10 2011 (comment link)

Thanks, Dave. That may be it. I don't lknow that I was thinking of anything. I was trying to determine what Bob Z. was referring to.

That option looks interesting, in any case. I see GUIDS. Wonder where that could lead.

Thanks, Rick

---

by Dave Harms on May 10 2011 (comment link)

It leads to disaster if you mess with the GUIDs :)

---

by Russell Eggen on May 10 2011 (comment link)

The new build is definitely an improvement. Still don't notice any differences in speed from 7.3, but all the nagging issues appear to be resolved (like not being able to use the letter "E" in some dialogs).

The only oddity I've spotted seems to be a permanent orphaned embed node. Used to be if you don't have orphans (caused by deleting a control with embed code attached to it), it never showed. Now it seems that whatever filter prevented it from showing is gone. No adverse effects from this, but might be a good spot for notes to self .

Still testing, no 3rd parties, no addons allowed while testing in motion. So far, so good. I've not come across anything yet that says "not ready for production",

Russ

by Richard Dafler on May 10 2011 (comment link)

Dave GUIDs,... still thinking about an article? How about a working title of "Beyond these GUIDs thar' be Dragons."? Thanks, Rick

by Dave Harms on May 10 2011 (comment link)

Rick, it's still on my list. I just haven't got there yet.

by Russell Eggen on May 11 2011 (comment link)

There does appear to be an issue with embeds, near as I can tell from what others are saying, embeds placed under a control (like a button) go to orphaned nodes. I've not seen this myself and I don't have enough information to replicate. Apparently SV has confirmed the issue so a fix should be in the next build.

Russ

by Richard Dafler on May 12 2011 (comment link)

My experience this morning was definitely something funny going on with embeds. All embeds in a prodedure would go to orphans. Close out and restart and they would be back. Do some work, check on the embeds,... all orphaned. Very strange and hard on the nerves.

Good to know S.V. is aware of it. Tnx, Rick Dafler

by Dave Harms on May 12 2011 (comment link)

The 8312 build has been pulled from the download site, perhaps because of the embed issues.

by Paul Blais on May 14 2011 (comment link)

While I have found embed issues the code still generates properly and compiles. I have one app where you get no emebeds from the screen formatter but it compiles and edits in the embeditor just fine. It is a bit unnerving working and knowing this problem is there.

Bob Z mentioned it in the ClarionLive webcast yesterday that the new ability to recall the the last embed point accessed has made some complex problems that have shown up as reported. At this point, I wouldn't start ripping apps apart to attempt a work around. The delay is mostly to sort out and find the full answer before they try another release. It's clearly high on the radar. I'm thinking it's all an IDE issue that will just go away with the fix.

by Stephen Mull on May 18 2011 (comment link)

SV is releasing the new release later today (May 18th) according to SV Sales, was just notified of this about an hour ago. --S

by Geoff Robinson on May 18 2011 (comment link)

new build is out - 8.0.8358

http://softvelocity.cachefly.net/C8/readme_8.0.8358.txt

has the details

cheers

Geoff R

---

by Dave Harms on May 25 2011 (comment link)

There's been a lot of discussion about IDE flicker and slowness. JP reports that he avoids this by renaming the clarion.exe.manifest file so the app can't find it.

---

by Dave Harms on May 25 2011 (comment link)

A note from Ben Dell on using the IP driver with C8:

> Quirk with Clarion 8 and IP Driver Server DLL Creation.
>
> If your Clarion Version against which you compile is Clarion 8, AND you create a NEW IP_dll.dll AND you create a server procedure that calls a program on the server with the RUN command AND you call ipx.Exec('serverprocedure') from your app , RESULT the IP Server throws a message 'Index out of Range'
>
> HOWEVER
>
> IF however you set the Clarion Version to compile @ Clarion 7 Templates AND you create a NEW IP_dll.dll AND you create a server procedure that calls a program on the server with the RUN command AND you call ipx.Exec('serverprocedure') from your app RESULT The IP Server is quite happy to keep on working and the external app is called and works as expected.
>
> There is no PTSS opened on this as we are still waiting for Clarion 8 IP Server / Driver Installs

---

by Dave Harms on May 25 2011 (comment link)

Günter Siegismund reports that on a TXA import the procedures get a modify date of today, instead of the original modification date.

Robert Barton has been carrying the torch on the new Report Writer, and got some replies from Bob Z regarding complex master/detail reports and memory usage. Z has indicated the memory usage issue is now being reviewed.

Phil Carroll points out that you can modify the start page styles by editing %root%\data\resources\startpage\Layout\default.css

> The class of the list layout is .copy.
>
> The class of the unselected headers (tabs) is .navi, and the selected one is .naviActiv.
>
> The white header text is .head.

Lee White, who always has to do things differently, does not use ClearType font smoothing if he can help it. But IE9 uses font smoothing and doesn't give you the option to disable it. The IDE uses the IE browser control to display the start page, so the way Lee disabled font smoothing for the start page was by downgrading to IE8.

A couple of users have reported a problem with disappearing LIB files, fixed by closing the solution, reopening, and regenerating. And presumably compiling.

---

by Dave Harms on May 27 2011 (comment link)

There have been a couple of reports on the IDE hanging when you go into the embeditor.

And while not a C8 news item, we will hopefully see a Clarion.NET release this weekend.

BACK TO TOP

Tip of the Week: Mass Control Changes

By Dave Harms

Posted May 25 2011

As in C6, you can make mass changes to controls in C7. In Figure 1 I've held down the shift key and have selected five different controls, three string controls and two entry controls.

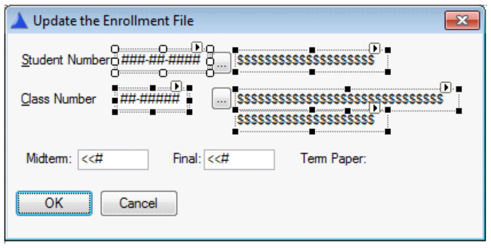
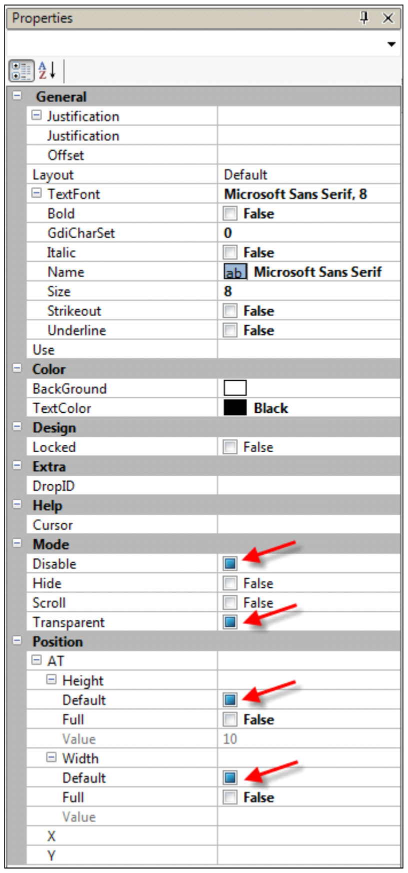


Figure 1. Selecting multiple controls

Figure 2 shows the Property Pad after those controls have been selected.



Properties

General	
⊟ Justification	
Justification	
Offset	
Layout	Default
⊟ TextFont	**Microsoft Sans Serif, 8**
Bold	☐ **False**
GdiCharSet	**0**
Italic	☐ **False**
Name	ab **Microsoft Sans Serif**
Size	**8**
Strikeout	☐ **False**
Underline	☐ **False**
Use	
Color	
BackGround	□
TextColor	■ **Black**
Design	
Locked	☐ False
Extra	
DropID	
Help	
Cursor	
Mode	
Disable	▣
Hide	☐ False
Scroll	☐ False
Transparent	▣
Position	
⊟ AT	
⊟ Height	
Default	▣
Full	☐ **False**
Value	10
⊟ Width	
Default	▣
Full	☐ **False**
Value	
X	
Y	

Figure 2. The Properties Pad with multiple controls selected.

There a couple of things to note about Figure 2. First, only properties common to all the selected controls are displayed. Second, numeric or string values are only shown if they're the same for all selected controls. And finally, checkboxes can be in one of three states: unchecked for all, checked for all, or checked for some (as indicated by the red arrows).

Remember that you have multiple levels of undo in the designer - if you make a mistake just press Ctrl-Z to back up.

Incidentally, the "checked for some" condition is not what you get with the "three state checkbox" that's provided with C8. That third state is simply "unknown", although it seems to me that with latched icons you might be able to fake a "checked for some" appearance.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors (ASJA).

Article comments

BACK TO TOP

Time And Again

By Steven Parker

Posted May 27 2011

Time. Time and again it comes up on the news groups. Time. Sometimes the question is about comparing two times, that is, about time arithmetic. Those questions are quickly dispatched.

More often, indeed several times a year, the developer understands how to do time arithmetic in Clarion. What the developer does not understand is how to display the difference between two times.

See the bibliography, at the end of this article, for a selection of the articles available on the subject of time. If you aren't familiar with time computation, take a few moments to read "The Clarion Advisor – Calculating Times." I consider this article essential reading for anyone having to do any kind of time arithmetic.

My articles, "Marking Time, Part I" and II, present a number of time computing and displaying procedures. And, Kevin Erskine's "Another Approach to Time Calculations" is a thorough set of procedures for computing and displaying times.

Time Arithmetic

Clarion either stores or handles both dates and times as Longs. It stores them as Longs in Clarion DAT and TPS files. Or, in the RTL, it treats them as Longs, as in Date and Time data types. Or, it easily allows casting them as Longs, as with xBase dates.

Because Clarion stores/handles dates as Longs, date arithmetic is simple arithmetic. For example:

```
ElapsedDays = EndDate - BeginDate
TotalDays = EndDate - BeginDate + 1
FutureDate = DateVariable + numberOfDays
```

My favorite is:

```
xDate = Deformat(20040419,@d12)
If (Today() - xDate) % 14 = 0
  RecurringProcedure
End
```

This code allows me to perform `RecurringProcedure` every 14 days, precisely on the anniversary of 19 April 2004.

Likewise, because Clarion stores/handles time of day as a Long, time arithmetic is also very easy. But, because midnight is defined as "1" (there is no zero in Clarion time), there is an inherent offset of one (see "The Clarion Advisor - Calculating Times" for a complete discussion of handling the issues this can and does cause):

```
ElapsedTime  = (Clock() - 1) - (BeginTime - 1) + 1
```

And, because the result of a time computation is a numeric, computing and displaying elapsed time or displaying a countdown timer is *just* arithmetic. See "Marking Time, Part 1," "Marking Time: Round 2" and "Another Approach to Time Calculations" to see various approaches to computing and display time differences.

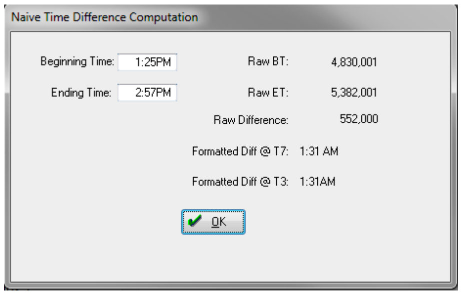
Time computation is a very popular subject in the Clarion world. I have cited four articles but there are more. And despite this fairly extensive coverage, questions about computing the difference between two dates and times recur on the newsgroups with almost predictable regularity.

The predictable, several times a year problem is: Now that I have computed the difference between the times, I cannot display it using a “Time Picture” correctly.

The Problem

Suppose I have a beginning time of 13:25 and an ending time of 14:57. The difference is one hour and 32 minutes (1:32).

If I subtract the beginning time from the ending time, in the standard way, and display the result in a Time Picture field, I get this:



Naive Time Difference Computation

Beginning Time: 1:25PM
Ending Time: 2:57PM
Raw BT: 4,830,001
Raw ET: 5,382,001
Raw Difference: 552,000
Formatted Diff @ T7: 1:31 AM
Formatted Diff @ T3: 1:31AM

Figure 1. The “problem” illustrated

There is a test app, `NaiveTimeComp`, at the end of this article. Open it up and you can follow along. “Beginning Time” and “Ending Time” are @T7, so your local time convention will be honored.

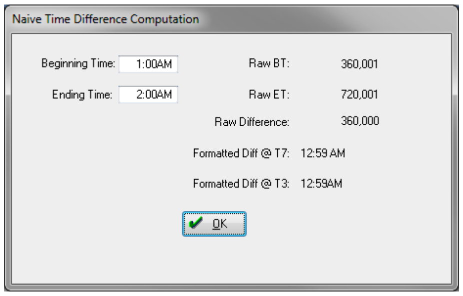
“RawBT” and “RawET” are the beginning time and ending time fields, respectively, but displayed as the Longs that the RTL thinks they are. “Raw Difference” is the result of simple arithmetic.

There are two display fields, using different time pictures. But, neither displays anything that can be interpreted as “one hour, 32 minutes.” Work with the app (it is a 7.3 app). Nothing you can do with Time Pictures will give you 1:32 or anything close.

But, the displayed value, “1:31 AM,” seems close enough that it is tempting to think I’m close to the solution.

Pure serendipity. And quite wrong.

Run the app again. This time enter “1” (0100) and “2” (0200). The difference is one hour. Instead, you will see this:



Naive Time Difference Computation

Beginning Time: 1:00AM
Ending Time: 2:00AM
Raw BT: 360,001
Raw ET: 720,001
Raw Difference: 360,000
Formatted Diff @ T7: 12:59 AM
Formatted Diff @ T3: 12:59AM

Figure 2. The “problem” illustrated

I do not see how “12:59 AM” can possibly be confused with something close to what I want. (You will get this result whether you enter the single digits “1” and “2” or the complete time, down to the second, “010000” and “020000.”)

The reason is simple. The reason I can’t get “one hour” out of

```
Format(RawDifference,  @T7)
```

is that Time Pictures are specifically designed to display data as “time of day.” I do not want the time of day here. I want “elapsed time,” which RawDifference does contain, but “T” pictures just aren’t intended for nor are they adaptable to showing elapsed times.

The Solution

The solution is fairly simple. Once you realize that `RawDifference` is not a time (it represents an amount of time, not **a** time) and, therefore, give up on Time Pictures and treat time variables as simple numbers, it *is* simple.

"Time" in Clarion is a number between 1 and 8,640,000, each integer representing 1/100 second. And, if Clarion measures time in increments of 1/100 second (even when the computer does not measure that granularly), there are some basic equations I can derive:

```
1 second = 100
```

therefore,

```
1 minute = 6000 (i.e., 60 * 100)
1 hour = 360,000 (i.e., 60 * 6000)
```

etc. And I now have a number of ways to proceed.

In that first article, “Marking Time, Part I,” I created a library of procedures to calculate:

- the number of seconds between two times
- the number of minutes between two times
- the number of hours between two times
- the elapsed time
- when to initiate a timeout
- how much time remains before timeout
- whether or not time is up.

At that time, I noted that the first three procedures, how many seconds, minutes and hours, are used to construct the fourth, the elapsed time. So getting a displayable elapsed time shouldn’t be too hard.

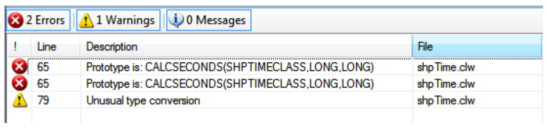
That Was Then, This Is Now ...

ABC and OOP extensions for Clarion were still in the future at the time I created my “ET” functions (the first of them were created for CPD, Clarion Professional Developer, DOS). By the time of CDD and Clarion for Windows, until C4, that meant that function libraries were maintained as LIBs or as DLLs (or both). One app for each target was required. Or, to reuse the functional code, one PRJ and one EXP was required for each target (oh, yes, 16 bit was also supported at the time).

In the immortal words of the inestimable Pooh, “bother!”

But, as I discovered in my recent construction of a string class, Classes are an appropriate replacement for the function libraries, the LIBs and DLLs of old (and classes don’t care whether the app produces a LIB, a DLL or an EXE). In fact, in “A Class for Tagging” (Clarion Mag, Feb 2003), I realized that I could use my existing code pretty much “as is.” “Pretty much,” meaning that there were necessary adaptations to the syntax required by class structures. But few modifications to code were needed.

These adaptations include, especially because I constantly forget, including the class’ Label in all the procedure definitions in the CLW. Forgetting to do so results in compiler errors, errors that make it look like I called the method incorrectly. In fact, the compiler seems to be constructing the prototype as I *should have* typed it:



!	Line	Description	File
✖	65	Prototype is: CALCSECONDS(SHPTIMECLASS,LONG,LONG)	shpTime.clw
✖	65	Prototype is: CALCSECONDS(SHPTIMECLASS,LONG,LONG)	shpTime.clw
⚠	79	Unusual type conversion	shpTime.clw

Figure 3. Forgetting to include class Label in procedure definition

“Bother!” indeed.

But, as I noted in that article, the class declaration, the INC file, is just a modified version of the same Map Include File I’d been creating since making LIBs and DLLs got easier in CDD, DOS. The INC file is just the existing Map Include File plus the lines in bold face:

```
OMIT('_EndOfInclude_',_timing_)
_timing_ Equate(1)
shpTimeClass   CLASS,TYPE,MODULE('shpTime.clw'),LINK('shpTime.clw')
Construct               PROCEDURE
Destruct                PROCEDURE
TimeRemaining           FUNCTION(Long),String
IsTimeUp                FUNCTION(Long),Short
CalcEndTime             FUNCTION(Real),Long
CalcSeconds             FUNCTION(Long,Long),String
CalcMinutes             FUNCTION(Long,Long),String
CalcHours               FUNCTION(Long,Long),String
CalcET                  FUNCTION(Long,Long),String
              End
_EndOfInclude_
```

The Procedure Definition file, the CLW, similarly is – remember to include the class name! – pretty much the existing source files. The only other thing to remember is that if I need to call another procedure, I need to do *that* differently.

In a LIB or DLL, if I need to call another procedure I just make a standard procedure call:

```
If pTimeOutTime
  Hours = CalcHours(Clock(),pTimeOutTime)
```

Inside a class, this becomes:

```
If pTimeOutTime
  Hours = Self.CalcHours(Clock(),pTimeOutTime)
```

The Procedures

The purpose of the class, like the original function library, is to display certain time-related information to the end user. Specially, the things that I want to display are:

- when you will be timed out
- how long you’ve been working

and

- how long until you are timed out

I label these procedures CalcEndTime, CalcET (for “elapsed time”) and CalcTimeRemaining, respectively.

Additionally, I obviously need, above and beyond the two “display” functions, several procedures that are purely computational (I consider the `CalcET` procedure purely computational, since I may compute it for record keeping and not display to the user).

The core computations are the number of hours, the number of minutes and number of seconds between two times. There is more to it than expressing

```
EndingTime – BeginningTime
```

as total seconds, minutes or hours. Expressing the difference in this way does not provide me with the displayable information I want. In Figure 1, above, the total number of minutes between the beginning and ending times is 92 (one hour, 32 minutes is 92 minutes). If I am displaying "Elapsed time = *hh:mm:ss*," the number of minutes I want is "32," not 92.

Similarly, Figure 2 shows a difference of one hour. I do not want "60 minutes" as the number of minutes, I want "0." I want to display "Program use time: 1 hour, 0 minutes …," not "Program use time: 0 hours, 60 minutes …."

What I figured out when I first wrote these procedures was that I want to compute the difference between two times, then I want to "remove" the time blocks from the top down. I compute the number of hours first, then the number of minutes and, lastly, the number of seconds. So,

CalcHours is first.

CalcHours takes two parameters: BeginningTime and EndingTime. It returns a string containing the (integer) number of hours. Its prototype is:

```
CalcHours   FUNCTION(Long  pBeginTime,Long pEndTime),String
```

The code is simplicity itself (not in the included CLW; that was written years ago and involves a lot of copy and paste of extra variables and computations):

```
Difference = pEndTime –  pBeginTime
Hours = INT(Difference / 360000)
Return Hours
```

At the top of the CLW, I eventually included equates for all standard time periods:

```
eNoTim            Equate(0)
eMidnight         Equate(1)
eDay              Equate(8640000)
eTick             Equate(1)
eSecond           Equate(100)
eMinute           Equate(6000)
eHour             Equate(360000)
```

This prevents typos (not that I've ever typed 864000 when I meant 8640000). So this code could now be rendered:

```
Hours = INT(Difference / eHour)
```

This is the easiest because I only need one calculation: If I divide Difference by 360000 (the number of ticks in an hour), the result is a fractional number (a real). The integer portion represents the hours in Difference and the remainder represents the minutes and seconds. I do not need that remainder here (though I will in other time calculations).

CalcMinutes is next.

In `CalcHours`, I realized that dividing `Difference` by `eHour` left me with a fraction representing the number of minutes and seconds in `Difference`. Therefore,

```
MinSec = Difference % 360000
```

extracts minutes and seconds from `Difference`. I can use a strategy similar to what I used in `CalcHours` to get just the minutes:

```
Minutes = Int( MinSec / 6000 )
```

or

```
Minutes = Int( MinSec / eMinute)
Return(Minutes)
```

`CalcSeconds` is last. It employs substantially the same strategy as `CalcHours` and `CalcMinutes`. The key line being:

```
Secs = MinSec % 6000
```

The only substantive difference is that Clarion allows me to compute and return 1/100ths of a second:

```
If BeginTime
  Hundredths = Secs % 100
Else
  Hundredths = 0
End
```

Writing the `CalcET` procedure, the elapsed time, to display to the user is now fairly straightforward. `CalcET`, though I didn't originally write it that way (and, because it continues to work, haven't changed in the CLW), is just stringing together the methods already written:

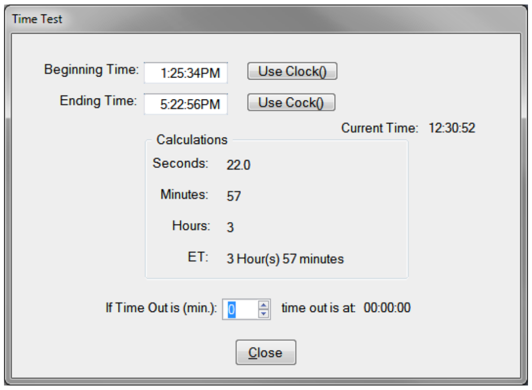
```
Hours   = Self.CalcHours(EndTime , BeginTime)
Minutes = Self.CalcMinutes(EndTime , BeginTime)
Seconds = Self.CalcSeconds(EndTime , BeginTime)
Seconds = Secs / 100
TimeString = Clip(Hours) & ':' & Clip(Minutes) & ':' & |
     Clip(Seconds) ! alternate returnable
LongTimeString = Clip(Hours) & ' Hour(s) ' & |
Clip(Minutes) & ' minutes ' & |
     Clip(Seconds) & ' seconds'
Return(LongTimeString)
```

If I want the individual components, hours, minutes, seconds, I have a number of choices.

- I can call the methods individually, assigning to my target variables
- I can change the `CalcET` procedure to return `TimeString` instead of `LongTmeSting` and parse it, if necessary, in the calling procedure
- I can, as Kevin Erskine did in his article, pass in a named group to receive the component values

Choices, choices ….

In the demo app, `TimeClassTest`, enter a beginning time and an ending time:


Time Test

Beginning Time: 1:25:34PM
Ending Time: 5:22:56PM
Current Time: 12:30:52

Calculations
Seconds: 22.0
Minutes: 57
Hours: 3
ET: 3 Hour(s) 57 minutes

If Time Out is (min.): 0 time out is at: 00:00:00

Figure 4. TimeClass Methods in Action

and you will see the individual computations as well as the elapsed time display string. (Note the entry fields are @T7 but there is no MASKing. Keep that in mind as you enter digits.)

So, my class computes seconds, minutes and hours. It can display an elapsed time. I need three more methods. I need to:

- Compute the time at which I want to time a user out (also useable as in inactivity timer)
- If there is a time out time, has that time been reached
- If there is a time out and I want to display “time remaining,” I need to compute that.

CalcEndTime is easy. For me, “time to allow” (or “inactivity period to allow”) is expressed in number of minutes. Therefore:

```
CalcEndTime          FUNCTION (Real pTimeAllowed)
  CODE
  Return( Clock() +  (pTimeAllowed * eMinute) )
```

is all I need. (In Figure 4, this is found on the last line. Note that it uses the current date and time, no the beginning or ending time.)

IsTimeUp is also routine, assuming I have a variable holding the time of day at which the user is to be timed out (that is, I previously called CalcEndTime):

```
IsTimeUp             FUNCTION (Long pTimeOutTime)
  CODE
  If pTimeOutTime > Clock() or ~pTimeOutTime
    Return(False)
  Else
    Return(True)
  End
```

Note how the method can be safely called if a time out time has not been set.

Where my class instance is called Mytimer, using TakeEvent:

```
If Event() <> Event:Timer
  TimeOutTIme =  MyTimer.CalcEndTime(cfg:MinutesAllowed)
End
```

coupled with:

```
If MyTimer.IsTimeUp(TimeOutTime)
  Post(Event:CloseWindow)
End
```

become an effective inactivity timer. (For every non-Timer event, i.e, keyboard or mouse event, the time out time is reset.)

The final item needed is a way to show the user how much time s/he has left. The TimeRemaining procedure calls the three base methods, concatenates a display string and sends it back to the caller:

```
TimeRemaining     FUNCTION (Long pTimeOutTime)
ReturnString         STRING(20)
Hours                LONG
Minutes              LONG
Seconds              LONG
  CODE
  If pTimeOutTime
    Hours   = Self.CalcHours(Clock(),pTimeOutTime)
    Minutes = Self.CalcMinutes(Clock(),pTimeOutTime)
    Seconds = Self.CalcSeconds(Clock(),pTimeOutTime)
    If Hours
```

```
      ReturnString = Clip(Hours) & ':'
    End
    ReturnString = Clip(ReturnString) & Clip(Minutes) & |
      ':' & Clip(Seconds)
  Else
    ReturnString = ''
  End
  Return(ReturnString)
```

In DOS, I placed a string on the screen to show this. In Windows,

```
0{Prop:StatusText,2} = |   MyTimer.TimeRemaining(LOC:TimeOutTime)
```

works very nicely.

Summary

Well, converting my old procedure libraries to classes turns out to be quite easy. Because I don't have to maintain multiple .APPs or multiple .PRJs and .EXPs, I think converting them to classes is actually desirable.

Except for one minor item, this time class deals with the need to display times to a user. So we now all have a ready answer the next time the question comes up on a news group.

The "one minor item?" These methods make no provision for "midnight rollover," for start and end times being on different days. That subject … soon.

Bibliography

- Dave Harms, The Clarion Advisor – Calculating Times
- S.H.Parker, Marking Time, Part 1
- S.H, Parker, Marking Time: Round 2
- Kevin Erskine, Another Approach to Time Calculations
- S.H. Parker, Throwing Users Out: Methods of Computation
- S.H Parker, Replicating IDLE: All Quiet on the Keyboard?

Download the source

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since version 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.

Article comments

by Geoff Robinson on June 1 2011 (comment link)

Hi Steve

thanks for the article - a couple of comments:

you say

ElapsedTime = (Clock() - 1) - (BeginTime - 1) + 1

when calculating the difference between two times, the fact that times start at 1 and not zero is irrelevant - you just want the difference between the two times in hundredths of seconds so the correct way to do it is simpler:

ElapsedTime = clock() - BeginTime

so, more generically, just as you say

ElapsedDays = EndDate - BeginDate

for time you would say

ElapsedTime = EndTime - BeginTime

also PLEASE try to avoid all the unnecessary clips(). They are not required and make the code really inefficient.

For example:

```
If Hours
  ReturnString = Clip(Hours) &amp; ':'
End
ReturnString = Clip(ReturnString) &amp; Clip(Minutes) &amp; |
  ':' &amp; Clip(Seconds)
```

has four clips - all of them unnecessary. Instead try:

```
ReturnString = choose(~Hours,'',Hours &amp; ':') &amp; Minutes &amp; ':' &amp; Seconds
```

much nicer!

regards

Geoff R

---

by Steven Parker on June 1 2011 (comment link)

Geoff:

> ElapsedTime = (Clock() - 1) - (BeginTime - 1) + 1 I thought so too. Right until some of the code came up 1 tick off -- so I just changed it everywhere (it's probably just the "+1" that was needed.

As to the CLIPs, this is very old code .

BACK TO TOP

CLARION MAGAZINE
READ. LEARN. SOLVE.

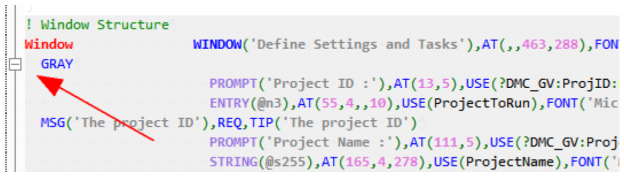
Home | Subscribe | E-Books | News | Blog | Store | My ClarionMag | My Lists | Contact

Custom Code Folding

By Russell Eggen

Posted May 28 2011

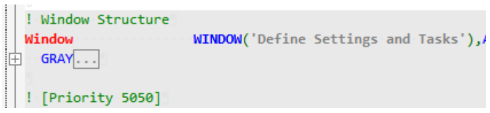
Starting with Clarion 7 and continuing into Clarion 8, the concept of code folding is now part of the Clarion toolkit. Figure 1 shows what foldable code looks like in the embeditor:



```
! Window Structure
Window                  WINDOW('Define Settings and Tasks'),AT(,,463,288),FON
  GRAY
                          PROMPT('Project ID :'),AT(13,5),USE(?DMC_GV:ProjID:
                          ENTRY(@n3),AT(55,4,,10),USE(ProjectToRun),FONT('Mic
  MSG('The project ID'),REQ,TIP('The project ID')
                          PROMPT('Project Name :'),AT(111,5),USE(?DMC_GV:Proj
                          STRING(@s255),AT(165,4,278),USE(ProjectName),FONT('
```

Figure 1. Expanded code

The key is that little minus sign in the left hand margin. If you click on that minus sign, the code folds as shown in Figure 2.



```
! Window Structure
Window                  WINDOW('Define Settings and Tasks'),A
  GRAY...

! [Priority 5050]
```

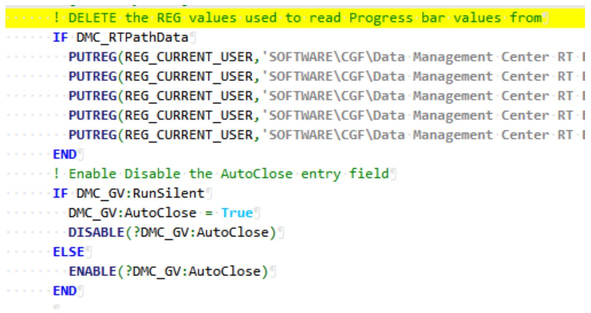
Figure 2. Folded code

The code is not gone, it’s simply hidden until you click on the plus sign and it’s now visible again. The effect is only cosmetic; the compiler works as it’s always done.

This is all fine and dandy, the only problem is that you are forced into someone else’s idea of what areas of code need folding and what don’t. For instance, some Clarion statements such as `IF` and `CASE` do not support code folding.

Adding your own code folding

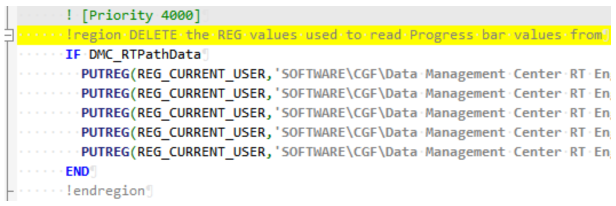
I’ll show to how to add custom code folding using a small piece of code in one of my own projects (Figure 3).



```
      ! DELETE the REG values used to read Progress bar values from
      IF DMC_RTPathData
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Center RT
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Center RT
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Center RT
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Center RT
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Center RT
      END
      ! Enable Disable the AutoClose entry field
      IF DMC_GV:RunSilent
        DMC_GV:AutoClose = True
        DISABLE(?DMC_GV:AutoClose)
      ELSE
        ENABLE(?DMC_GV:AutoClose)
      END
```

Figure 3. Code I want folded

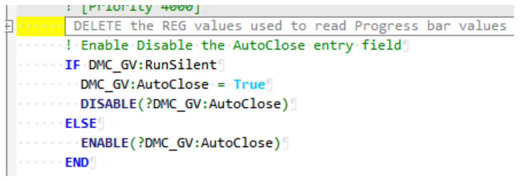
All you need to do is add a “magic comment.” The start of the fold begins with ! region and ends with ! endregion. You have the option of adding a label for the folding region. In the above example I decided to use the comment as the label (Figure 4)



```
      ! [Priority 4000]
      !region DELETE the REG values used to read Progress bar values from
      IF DMC_RTPathData
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Center RT En
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Center RT En
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Center RT En
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Center RT En
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Center RT En
      END
      !endregion
```

Figure 4. Region folding added

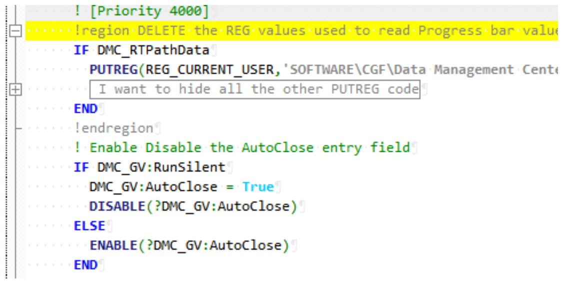
Notice the syntax colors change from an ordinary comment in green to gray. Click the minus sign and the code looks like Figure 5.



```
      ! [Priority 4000]
       DELETE the REG values used to read Progress bar values
      ! Enable Disable the AutoClose entry field
      IF DMC_GV:RunSilent
        DMC_GV:AutoClose = True
        DISABLE(?DMC_GV:AutoClose)
      ELSE
        ENABLE(?DMC_GV:AutoClose)
      END
```

Figure 5. Custom code folding in action

You may add as many folding regions as you like. This technique even works with nested regions as shown in Figure 6.



```
      ! [Priority 4000]
      !region DELETE the REG values used to read Progress bar value
      IF DMC_RTPathData
        PUTREG(REG_CURRENT_USER,'SOFTWARE\CGF\Data Management Cente
         I want to hide all the other PUTREG code
      END
      !endregion
      ! Enable Disable the AutoClose entry field
      IF DMC_GV:RunSilent
        DMC_GV:AutoClose = True
        DISABLE(?DMC_GV:AutoClose)
      ELSE
        ENABLE(?DMC_GV:AutoClose)
      END
```

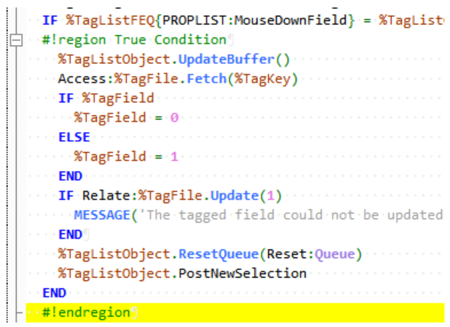
Figure 6. Nested region, folded.

Just be sure to use !endregion for each region you define.

What about templates?

This technique works in template code too, with some limitations. There is a slight variation required as a template comment requires two characters: `#!` This means a region for template code begins with `#! region <label>` and ends with `#!endregion`.

Figure 7 shows an example.



```
IF %TagListFEQ{PROPLIST:MouseDownField} = %TagList
#!region True Condition
  %TagListObject.UpdateBuffer()
  Access:%TagFile.Fetch(%TagKey)
  IF %TagField
    %TagField = 0
  ELSE
    %TagField = 1
  END
  IF Relate:%TagFile.Update(1)
    MESSAGE('The tagged field could not be updated
  END
  %TagListObject.ResetQueue(Reset:Queue)
  %TagListObject.PostNewSelection
END
#!endregion
```

Figure 7. Adding code folding in template code

Limitations in templates

Code folding using regions appears to only work inside target code (that is, code that will be generated as source) as Figure 7 shows. Adding this inside template-only code (for example #PREPARE structures) has no effect. This does seem to align with the inconsistent code folding support in the template language. So the benefit for template coders at first glance appears to be limited.

Summary

This technique has a side effect benefit; it adds a little more documentation to your code, which is always a good thing. I'm not aware of any limits to how deep nesting works (ten levels is no problem), but I admit to not going deeper than two levels in my own use. The benefits in template coding are limited, although the documentation benefits are not diminished. And template code is not self-documenting.

Russ Eggen has been using Clarion since 1986. Until about 1996, he was using it for business applications, mostly accounting programs. Afterwards he joined Topspeed as a consultant, and later as an instructor. He was a founding member of SoftVelocity when that company formed from Topspeed in May 2000. He left SoftVelocity in January 2001 and now works for his own company, RadFusion Inc. He still teaches and lectures, and is currently working on a new book and setting up a local Clarion classroom. Russ enjoys flying, scuba, and applied philosophy, and with great effort you might coax him into political discussions.

Article comments

by Terence Davidson on June 3 2011 (comment link)

Thanks Russ. Nice tip. Pity if/end and Loop/end aren't automatic folds.

by Russell Eggen on June 3 2011 (comment link)

Terence,

I agree! I think any statement (template or otherwise) that has an END statement in any form should be subject to code folding.

Russ Eggen

BACK TO TOP

Time And Time Again

By Steven Parker

Posted May 30 2011

Midnight rollover was not accounted for in the conversion of my old time libraries (see "Time and Again"). I have methods for computing time of day when a user should be timed out (`CalcEndTime`) and to compute the elapsed time between two times (`CalcET`). Neither of these methods takes midnight rollover into account.

And these methods do need to deal with the possibility that the start and end or start and time-out times are on opposite sides of a date line. In fact, examining the methods developed in `shpTimeClass`, these are the only two methods that seem to need it (thus, lightening my work load quite a bit).

Framing the Problem

When doing time computations, it is first necessary to understand *how* to determine whether one (or more) days have passed. It is not safe to simply assume that two times are on the same day. And, as I showed in the previous article, this assumption severely restricts the utility of time-computation code. Neither is it safe to assume that, if there has been a midnight rollover, only one day has passed. In fact, it isn't even safe to assume that there has been a midnight rollover at all.

No assumptions are safe. All circumstances must be accounted for and all circumstances must be tested.

I need four variables:

```
BeginDate
BeginTime
EndDate
EndTime
```

to hold the data I'm interested in. To do time math, testing for midnight rollover, all four of these data are used.

I want to be able to determine the difference between EndTime and BeginTime. So first, I need to determine whether or not one or more days have passed. This bit is simple enough, just compare `BeginDate` and `EndDate`.

There are exactly three possibilities:

- `EndDate > BeginDate`
- `EndDate = BeginDate`
- `EndDate < BeginDate`

If `EndDate = BeginDate`, the dates are the same, there has been no midnight rollover and my existing methods work, as is.

The remaining two conditions require more thought.

`BeginDate > EndDate` should never evaluate as true; this condition should never occur. If `BeginDate > EndDate`, I would think in terms of a very severe message and summarily terminating the application or, at least, breaking out of the computational cycle.

That leaves one condition, `EndDate > BeginDate`. In this case, at least one midnight rollover *has* occurred, Unfortunately, things aren't quite as straightforward as one might like.

While `EndDate - BeginDate` tells me how many calendar days have elapsed, the time variables cannot be directly compared any more. There may have been a midnight rollover but this does not mean that "a day" (24 hours) has elapsed. Neither does it mean that only one day has passed. This is obvious when considering a time clock.

Case 1: Suppose an employee clocks in at 23:00 and clocks out at 07:00. `EndDate` is greater than `BeginDate` but, in this case, `EndTime` is less than `BeginTime`. Instead of translating to eight hours, subtracting

```
ElapsedTime = EndTime - BeginTime
```

results in a negative number (-576000 ticks), an impossibility (time travel into the past is impossible, says Dr. Einstein). If I'm clever and use the absolute value or try to subtract the smaller number from the larger, I get 16 hours.

Why? 23:00 (11:00 P.M.) is 8,280,000 in Clarion native time (100th seconds since midnight, actually stored as 8,280,001). 07:00 is 2,520,000 clock ticks. The difference is -5,760,000 which, passing it through the various computational methods in `shpTimeClass`, translates back to -16 hours.

It should be eight, of course. And positive.

Case 2: The computations are even more bizarre for an employee clocking in at noon on Tuesday and clocking out at 17:00 Wednesday. Subtracting times gives seven hours instead of 29 (one day, five hours).

> Remember these two cases. They will be the test cases when I get to writing a `StarDated` elapsed time function.

If I add one day (8,640,000) to `EndTime`, Case 1 (the 23:00 to 07:00 case), and make the computation:

```
 11,160,000
- 8,280,000
===========
  2,880,000
```

I get a correct computation of eight hours.

So, a completely general codelet would look like:

```
If EndDate = BeginDate        ! same date
  ElapsedTime = (EndTime - 1) - (BeginTime - 1) + 1
Elsif EndDate < BeginDate          ! beginning date after ending
  Stop('Invalid dates!')
Else                     ! rollover occurred
  ElapsedDays = EndDate - BeginDate
  ElapsedTime = ((EndTime - 1) + (ElapsedDays * 8640000)) |
                - (BeginTime - 1) + 1
End
```

With a valid value in `ElapsedTime`, I can use the previously created methods. (I know this works, at least for a single midnight rollover; it's taken from some EOD (end of day) code I wrote … in CDD.)

In fact, if you are willing to deal with four variables, their storage and retrieval, I think this will handle all your time computing needs.

StarDates

StarDates are the easiest solution to the problem of cross-date time computations. A StarDate computation needs only two variables for computations, versus the four outlined above, making straight subtraction possible.

StarDating was introduced in the Logix Models for Clarion Professional Developer (about 20 years ago). A StarDate uses a single variable, typically a Real or Decimal (15,8) – I use one more digit than necessary as a container for rounding errors -- to contain both the date and the time. Therefore, two StarDates can be compared without worrying about midnight rollovers.

A StarDate is constructed by taking the Clarion standard date and adding the time expressed as a fraction. The integer portion of a StarDate is the date and the fractional portion is the time.

The time part can be expressed as a fraction by dividing it by 8640000 (the maximum number of clock ticks in a Clarion day). So a StarDate is constructed as follows:

```
StarDate = Today() +  ( Clock() / 8640000 )
```

StarDates are converted back into a date and a time by reversing the arithmetic.

```
StarDate =  Int(StarDate)
```

The time is the fractional portion and I get it by subtracting the date:

```
StarTime = StarDate -  INT(StarDate)
```

To finish the conversion to a Clarion standard time, multiply by 8640000.

```
StarTime *= 8640000
```

With a Clarion date and a Clarion time, I am able to use most of the time computation methods I've already created. But, more importantly, I'm back in the realm of the familiar. I have discrete and comprehensible variables and I can reuse my existing time class.

Reusing TimeClass: Inheritance

Most of the methods defined and tested in shpTimeClass, methods to compute number of seconds, minutes, hours, etc., look reusable here. Why copy them to a new class?

Why? Because my previous experiments with inheritance were a complete disaster.

A recent news group posting by Sean Cameron clarified inheritance for me. As it turns out, deriving a new class from a previously created class is easy.

In the INC file:

```
OMIT('_EndOfInclude_',_staring_)
_staring_ EQUATE(1)
 Include('shpTime.INC'),ONCE
shpStarClass   CLASS(shpTimeClass),TYPE,MODULE('shpStarClass.clw'),LINK('shpStarClass.clw
```

The changes to a standard class header file are in bold face. All the methods from the parent class, `shpTimeClass`, are available and useable in my new class. Thank you Sean.

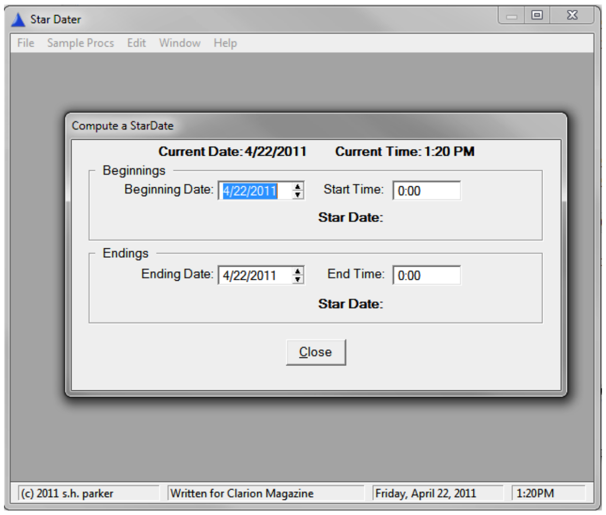
New Methods

Since I can reuse almost all my existing methods, I need to add only a few new methods. I need a method to create a StarDate out of a date and time. While I'm doing that, I might as well make a method to parse a StarDate back into its component date and time. Finally, I think I'll need a method to calculate the number of days between two dates.

MakeStarDate: this one is easy. Pass a date, a time and a variable to receive the StarDate;

```
shpStarClass.MakeStarDate PROCEDURE(Long pDate,Long pTIme,*Decimal pStarDate)
  CODE
   pStarDate = pDate + (pTIme / eDay)
```

In the demo app, Sample Procs | Compute a StarDate:



Current Date: 4/22/2011 Current Time: 1:20 PM

Beginnings
Beginning Date: 4/22/2011 Start Time: 0:00
Star Date:

Endings
Ending Date: 4/22/2011 End Time: 0:00
Star Date:

(c) 2011 s.h. parker | Written for Clarion Magazine | Friday, April 22, 2011 | 1:20PM

Figure 1: Compute a StarDate

You will notice that I actually provide two date and time fields in the demo procedure. This is due to having copied the procedure from one that did … more (and, which, I shall revisit shortly). But, this does allow you to enter two dates/times, with very close times, and compare the results.

`ParseStarDate`: also an easy one. Get the date from the integer portion of a StarDate passed into the method. Get the time by subtracting the integer portion and multiplying by "Clarion ticks in a day."

```
shpStarClass.ParseStarDate    PROCEDURE(*DECIMAL pStarDate, |
*LONG pDate, *LONG pTime)
    CODE
 pDate = INT(pStarDate)
 pTime = (pStarDate - pDate) * eDay
```

("eDay" -- remember the Equates I configured in "Time and Again.")

This method, if you check the INC file, I override. I provide a version for StarDaters who like Decimals and a version for those who prefer Reals. I suppose what I ought do is change Decimals and Reals, as I do seem to use them inconsistently, to "runtime defined" (?) parameters and let Clarion's automatic type conversion handle everything else for me.

`CalcDays`: this is a hard one.

This one is difficult because while two StarDates may contain different Date components, the two time stamps may be within a single 24 hour period. In these cases, while the date difference may be one, the time difference is less than `eDay` (8640000, Clarion "max" time). If less than `eDay`, I want a `CalcDays` method to return zero (0).

With that decision made, StarDates make this *so* easy. When the date portion has incremented by one but the end time is less than the starting time, normal arithmetic – straight subtraction -- will return a result less than 8640000. The method's code turns out to be quite simple:

```
shpStarClass.CalcDays    Procedure(*Decimal pBeginStarDate, |
 *Decimal pEndStarDate)    !,Long
Difference           DECIMAL(15,8)
    CODE
 Difference = pEndStarDate - pBeginStarDate
 Return Int(Difference)
```

In the demo app, Sample Procs | Compute "Days"

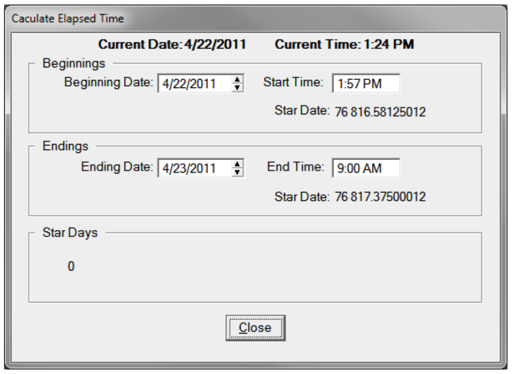


Figure 2: Compute Days

If, one other hand, I wanted a calendar day count:

```
shpStarClass.CountDays   Procedure(*Decimal pBeginStarDate, |
 *Decimal pEndStarDate)    !,Long
Difference           DECIMAL(15,8)
    CODE
 Return Int(pEndStarDate) – Int(pBeginStarDate)
```

would do the job.

Changed Methods

Two methods are affected by midnight rollovers, CalcEndTime and CalcET.

Because I feel I've pushed my luck far enough making Inheritance work, I don't want to get into virtualizing my class methods. So, I'm going to create new procedures.

CalcEndTime will be replaced by TimeUpTime. It will do the same thing as CalcEndTime, just under a new name, for StarDates (CalcEndTime, remember did not deal with midnight rollovers.. CalcET will be replaced by StarET.

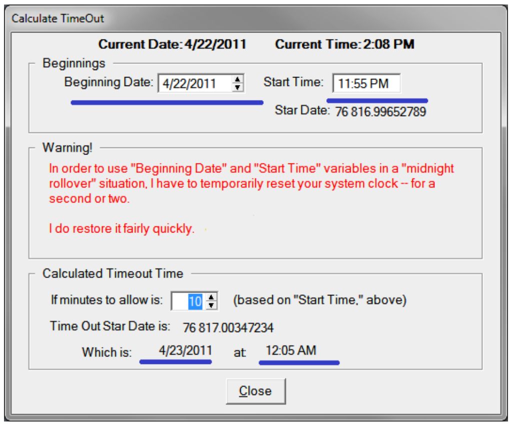
TimeUpTime: This procedure adds the passed number of minutes to Clock() to determine the time of day for timing out a user. It returns a StarDate.

If that addition is less than 8640000, the date portion is/remains Today() and the normal computation is still good. In that case, of course, the inherited method can be used.

If the addition results in a number greater than eDay, the date portion is/becomes Today() + 1 and the time portion is what is left after removing one day:

```
shpStarClass.TimeUpTime  Procedure(Real pTimeAllowed)  !,Real
X   LONG
TMP Real(15.8)      ! Star to be returned
Day Long
    CODE
 X = CLOCK() + (pTimeAllowed * eMinute)
 If X > eDay            ! computed time is after midnight
   Day = Today() + 1
   TMP = X - eDay
 ELSE                   ! same day
   Day = TODAY()
   TMP = Self.CalcEndTime(pTimeAllowed) ! call into parent class
 End
 TMP = Day + (TMP / eDay)
 Return TMP
```

WARNING: In the demo app, Sample Procs | Compute TimeOut (note, because the TimeOutTime method actually passes Clock(), the demo app has to reset your system clock for the duration of the method call, if you decide to test times across midnight):


Calculate TimeOut

Current Date: 4/22/2011   Current Time: 2:08 PM

Beginnings
Beginning Date: 4/22/2011   Start Time: 11:55 PM
Star Date: 76 816.99652789

Warning!
In order to use "Beginning Date" and "Start Time" variables in a "midnight rollover" situation, I have to temporarily reset your system clock -- for a second or two.

I do restore it fairly quickly.

Calculated Timeout Time
If minutes to allow is: 10 (based on "Start Time," above)
Time Out Star Date is: 76 817.00347234
Which is: 4/23/2011 at 12:05 AM

Close

Figure 3: Computing time at which to time user out

The window also uses the `ParseStarDate` method to provide a readable display.

`StarET`: This procedure is supposed to display the total elapsed time, the amount of time the user has been or was active. Given that I already had a functional elapsed time method, this should have been easy.

It wasn't. In fact, debugging it was a horror.

It *should* have been a matter of calling my newly created `CalcDays` method and using pretty much the same code and strategy as I did in `CalcET` (from the last article).

Same day timings worked just fine. The two test cases outlined earlier couldn't be made to work simultaneously. I'd get one case working, the other broke. Fix the second and another test case failed, which had worked 15 minutes before … "oh bother!"

That means that *I* probably misconceived something. I certainly started out by forgetting the code outlined at the beginning of this article, trying to be clever by using my existing methods.

I wonder; all existing methods, from `shpTimeClass`, are based on two times passed to the method. These parameters are in a known order, with known minimums and maximums. I wonder if inter-date data always honors these assumptions. Perhaps I should have made a method which takes a single numeric and extracts the hour, minute and second components.

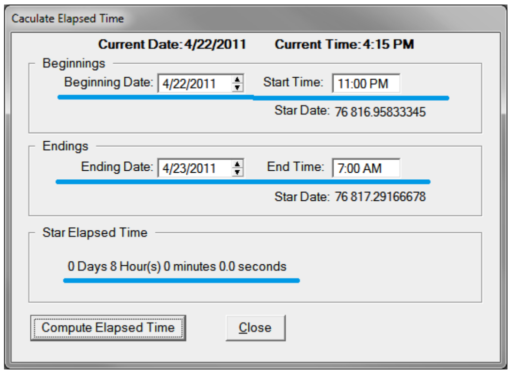
But, I hammered at the test cases until both worked. I ended up with hybrid code based on the code from the top of the article. My code mixes new and old but drops calls to existing methods for "new day" times. I left the old code, the code with which I struggled, in the CLW, OMITted, for you to play with.

The functional code is:

```
shpStarClass.StarET PROCEDURE(*Decimal pStarDateBegin,*Decimal pStarDateEnd)
Days                     Long
Difference               DECIMAL(15,8)
ET                       LONG
EndTime                  LONG
BeginTime                LONG
BeginDate                LONG
EndDate                  Long
Hours                    USHORT
MinSec                   LONG
Minutes                  BYTE
Secs                     LONG
Seconds                  BYTE
Hundredths               LONG
```

```
 TimeString           STRING(20)
 LongTimeString       STRING(60)
   CODE
  Self.ParseStarDate(pStarDateBegin,BeginDate,BeginTime)
  Self.ParseStarDate(pStarDateEnd,EndDate,EndTime)
  If BeginDate = EndDate
    Days = 0
    Hours = Self.CalcHours(BeginTime,EndTime)
    Minutes = Self.CalcMinutes(BeginTime,EndTime)
    Seconds = Self.CalcSeconds(BeginTime,EndTime)
  ELSIF EndDate >  BeginDate
    Days = EndDate - BeginDate
    !Days = Self.CalcDays (pStarDateBegin,pStarDateEnd)
    Difference = ((EndTime - 1) + (Days * eDay)) - |
 (BeginTime - 1) + 1
    BeginTime += BeginDate * eDay        ! one massive time number
    EndTime += EndDate * eDay           ! and, another one
    Hours = self.CalcHours(BeginTime,EndTime)
  ! stop('altn. days calc ' & INT(Hours / 24))
    If Hours > 24                        ! subtract out whole days
      Hours -= 24 * Days
    End
    Difference = pStarDateEnd - pStarDateBegin
    Days = Int(Difference)
    ET = (EndTime - 1) - (BeginTime - 1) + 1
    MinSec = ET % 360000
    Secs = MinSec % 6000
    Seconds = Secs / 100
    Minutes = Self.CalcMinutes(BeginTime,EndTime)
    !Seconds = Self.CalcSeconds(BeginTime,EndTime)
  End
```

Note how I finally took a cue from the discussion, earlier, and tested for `EndDate =` (or `<>`) `BeginDate`. In the demo app, Sample Procs | Compute ElapsedTime:



Caculate Elapsed Time

Current Date: 4/22/2011   Current Time: 4:15 PM

Beginnings
Beginning Date: 4/22/2011   Start Time: 11:00 PM
Star Date: 76 816.95833345

Endings
Ending Date: 4/23/2011   End Time: 7:00 AM
Star Date: 76 817.29166678

Star Elapsed Time
0 Days 8 Hour(s) 0 minutes 0.0 seconds

Figure 4: Elapsed StarDate time

Note that for this procedure, you have to push a button to get the computations to compute and display. During testing, I found the iterative calling of `TakeEvent`, which would have automated the display, very annoying.

Summary

Well, midnight rollover, time computations across multiple dates, is now class-ified. The `StarET` method was hardly as easy to write as I had thought it would be. But it does work and, in a class, allows inter-date time computations worry free.

I suspect that I treated `StarDates` a bit too much like standard dates and times.

As I said, perhaps a method that, borrowing the “one massive [time] number” cited in the `StarET` code, that extracts the time elements from (`StarDateEnd – StarDateBegin`) directly ….

Download the source

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since version 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.

Article comments

by surfersteve on June 5 2011 (comment link)

thank you very much for your efforts, as an tansaction accountant and tax data modeller im an glad someone has a grasp of these matters. if i place this in my code base i feel much gratitude. i owe so much to so many.

BACK TO TOP