Clarion Magazine

Clarion News

- » SoftVelocity Changes Release Policy
- » SuRF Demos File Schema Pad
- » DMC 1.5.0.3
- » J-Html 2.01 For C7 AppGen
- » StrategyOnline Price Increases Postponed To Nov 4
- » Clarion Third Party Profile Exchange October 21 2008 Update
- » Clarion Folk Ad Post Summary
- » SetupBuilder 6.9 Build 2390
- » DMC Birthday Special 25% Off
- » DMC 1.5.0.2
- » HCUG Meeting
- » CapeSoft "C7 Is Coming" 20% Off Special
- » What would you do?
- » Clarion Desktop 4.20
- » BFlat 1.0
- » vuMail 2.0
- » Noyantis CalendarPro Wrapper Template 1.09
- » Combit 20% Off
- » Clarion Third Party Profile Exchange Web Update Release
- » SetupBuilder 6.9 Build 2380
- » Clarion Folk Podcast
- » CapeSoft Special On The Way?
- » J-Cal Price Increase Deadline Extended
- » SimDatesClass Updated
- » Whitemarsh Data Semantics Management Book
- » EasyListPrint 1.15
- » CoolFrames 1.18
- » Noyantis Codejock TaskPanel Wrapper
- » SimDatesClass 1.03
- » amazingGUI 1.1.1
- » EZChangeLog Sneak Preview
- » BSec 1.02
- » DynaLib 4.1.5
- » vuMail 2.0 Feature List

[More news]

- » Clarion.NET FAQ
- » Clarion# Language Comparison

[More Clarion & .NET]

[More Clarion 101]

Latest Free Content

- » First Look: AppGen
- » AppGen Released!

Latest Subscriber Content

AppGen Beta Status Report

The first release of the C7 AppGen is now in beta testing with third party vendors. Dave Harms provides an update on bugs, fixes, features and the overall status of the beta.

Posted Friday, October 31, 2008

Understanding The C7 Build System

Clarion 7 introduces a powerful new build system capable of handling multi-app solutions and much more. Dave Harms looks at the basic concepts of solutions and projects.

Posted Thursday, October 30, 2008

Trading Global Data for Data Obfuscation, Sort Of...

Popular opinion to the contrary, global data is (or at least can be) a good thing. You wouldn't get far without it. Like any data you create, it should be used when it is the best fit for the need at hand, and with thoughtful consideration of its implementation and access methods. Mark Geisinger presents a class for managing multi-APP global data in a way that doesn't require you to recompile all your apps every time the data definition changes.

Posted Thursday, October 23, 2008

First Look: AppGen

Dave Harms takes a first look at the new AppGen, just released to third party vendors for compatibility testing. Free access article.

Posted Wednesday, October 22, 2008

AppGen Released!

It's been a long, long wait. And for a small group of Clarion developers, at least, that wait is now over. SoftVelocity today released C7 with the Application Generator to third party vendors for compatibility testing. Stay tuned for a First Look...

Posted Tuesday, October 21, 2008

Filtering Reports With The Pause Control

The Pause control is good for more than just pausing reports. As Steve Parker shows, with a little tweaking you can also use this control for report filter input.

Posted Tuesday, October 21, 2008

IP-Enabling An Existing Application

SoftVelocity's IP Driver replaces direct file access across a network with a communication protocol and a customized data server application. This reduces network traffic and increases data file stability. Dermot Herron shows how to IP-enable the People application. UPDATED Oct 17, 2008 with further info on ERROR(27).

Posted Friday, October 17, 2008

AppGen Clears Another Hurdle

Z has blogged about the successful resolution of the deeply nested window problem in the new IDE, which means that the AppGen is expected to go to third party developers "within the next few days," and to beta program participants shortly after.

Posted Thursday, October 09, 2008

More Clone(File)

In the last action packed episode, Dr. Parker showed how to clone file structures with LIKE. But this approach runs into problems if you want to preserve KEY structures. The solution: use a template.

Posted Tuesday, October 07, 2008

Source Code Library 2008.09.30 Available

The Clarion Magazine Source Code Library has been updated to include the latest source. Source code subscribers can download the August/September 2008 update from the My ClarionMag page. If you're on Vista please run Lindersoft's Clarion detection patch first.

Posted Wednesday, October 01, 2008

[Last 10 articles] [Last 25 articles] [All content]

Source Code

The ClarionMag Source Code Library

Clarion Magazine is more than just a great place to learn about Clarion development techniques, it's also home to a massive collection of Clarion source code. Clarion subscribers already know this, but now we've made it easier for subscribers and non-subscribers alike to find the code they need.

The Clarion Magazine Source Library is a single point download of all article source code, complete with an article

- » AppGen Clears Another Hurdle
- » Source Code Library 2008.09.30 Available

[More free articles]

Clarion Sites

Clarion Blogs

cross-reference.
More info • Subscribe now

Printed Books & E-Books

E-Books

E-books are another great way to get the information you want from Clarion Magazine. Your time is valuable; with our e-books, you spend less time hunting down the information you need. We're constantly collecting the best Clarion Magazine articles by top developers into themed PDFs, so you'll always have a ready reference for your favorite Clarion development topics.

Printed Books

As handy as the Clarion Magazine web site is, sometimes you just want to read articles in print. We've collected some of the best ClarionMag articles into the following print books:

- » Clarion Tips & Techniques Volume 4 - ISBN 978-0-9784034-09
- » Clarion Tips & Techniques Volume 3 - ISBN: 0-9689553-9-8
- » Clarion 6 Tips & Techniques Volume 1 - ISBN: 0-9689553-8-X
- » Clarion 5.x Tips and Techniques, Volume 1 - ISBN: 0-9689553-5-5
- » Clarion 5.x Tips and Techniques, Volume 2 - ISBN: 0-9689553-6-3
- » Clarion Databases & SQL - ISBN: 0-9689553-3-9

We also publish Russ Eggen's widely-acclaimed Programming Objects in Clarion, an introduction to OOP and ABC.

From The Publisher

About Clarion Magazine

Clarion Magazine is your premier source for news about, and in-depth articles on Clarion software development. We publish articles by many of the leading developers in the Clarion community, covering subjects from everyday programming tasks to specialized techniques you won't learn anywhere else. Whether you're just getting started with Clarion, or are a seasoned veteran, Clarion Magazine has the information *you* need.

Subscriptions

While we do publish some free content, most Clarion Magazine articles are for subscribers only. Your subscription not only gets you premium content in the form of new articles, it also includes all the back issues. Our search engine lets you do simple or complex searches on both articles and news items. Subscribers can also post questions and comments directly to articles.

Satisfaction Guaranteed

For just pennies per day you can have this wealth of Clarion development information at your fingertips. Your Clarion magazine subscription will more than pay for itself - you have my personal guarantee.

Dave Harms

ISSN

Clarion Magazine's ISSN

Clarion Magazine's International Standard Serial Number (ISSN) is 1718-9942.

About ISSN

The ISSN is the standardized international code which allows the identification of any serial publication, including electronic serials, independently of its country of publication, of its language or alphabet, of its frequency, medium, etc.

Clarion Magazine

Clarion News

Search the news archive

ClarioNET 6.0302

ClarioNET 6.0302 is now available for download. Changes include: Fixed listbox hanging; Fixed Droplists bug; Fixed DropCombo bug; Fixed Graphics; Fixed report preview bug (ABC); Fixed download via HTTP bug; Fixed Maximise frame bug; Double click (new bug) - some reports of failure still coming in (therefore still work in progress but released); ClarioNet popup selector calendar available from 3rdParty source; Returns to correct Control when returning from server. Please note that as this is the first official release in a very long time and is being made available to all ClarioNET users who have purchased ClarioNET since Ivan Mintoff took over. As from the next release, releases will be made to those falling within the annual maintenance agreement.

Posted Monday, November 03, 2008

ABCFree Templates And Tools Updated

Changes to the ABCFree Templates and Tools include C7 compatibility and miscellaneous minor changes. The installer has not yet been updated to work with the Clarion 7 directory structure.

Posted Monday, November 03, 2008

List & Label 14

List & Label 14 is now available. There are many new features including integrated real-time preview in the Designer.

Posted Monday, November 03, 2008

GTL6.41

GTL6.41 is now available from the Par2 download center. This release fixes a problem with starting Clarion twice when clicking on "Start Clarion."

Posted Monday, November 03, 2008

BFlat 1.2

BFlat 1.2 is now available, for C4-C7 Legacy and ABC. Changes include: Removed SDI restriction on Looks Good stuff; Added transparent controls; Added wallpaper (overrides color); Added buttons image replacement; Added Freeze checkbox on procedure extension so doesn't get updated; Overrides on all features on all procedures; Added "BoCap Looks Good" to global; and more. Demo available.

Posted Monday, November 03, 2008

DMC Gold 1.5.0.4

Posted Monday, November 03, 2008

Free Mallorca Collection Subset

1st Logo Design has released a subset of the Mallorca collection for free. Also includes seven toolbar Web2.0 backgrounds.
Posted Monday, November 03, 2008

Huenuleufu Templates For C7

The Huenuleufu templates have been tested against C7. FinalStep needs a minor change to be compatible in Clarion 7. The update release (version 2.20) is available on demand for current users no extra charge. FullRecord / FullRecord 2 works as it is in C7. The NeatMessage templates work perfectly in C7, but a bug was found in the run-time code that makes the message box text look wrong. SV acknowledged and hopefully there will be a fix soon. PrintWindow works as it is in C7. WindowID works as it is in C7.
Posted Monday, November 03, 2008

Freeware Locus Templates C7 Compatible

The Freeware Locus Templates are Clarion 7 compatible and works as expected (ABC and Legacy). A new build of the Locus template set for use by C7 AppGen will be released early next week.
Posted Monday, November 03, 2008

Noyantis RSS Feed

An RSS feed has been added to the Noyantis site to make it easier for people to be aware of new releases / developments etc.
Posted Monday, November 03, 2008

Noyantis CalendarPro 1.10 Beta

Version 1.10 Beta of the Noyantis CalendarPro wrapper template has been released. Modifications include: Recurring Events facility added (drag / resize not supported yet); DatePicker control enhanced to work standalone (eg, not connected to a calendar); Week Numbers facility added to DatePicker control; Date Limits added to DatePicker control; ClearViewSchedules method was causing a GPF when not in Day view - fixed. The new version can be downloaded from the Members area using the original download and registration details contained in your sales emails.
Posted Monday, November 03, 2008

SoftVelocity Changes Release Policy

SoftVelocity has changed its release policy. Formerly builds were first released to third party vendors for compatibility testing. Now such "pre-releases" will be made available to all subscription holders.
Posted Thursday, October 23, 2008

SuRF Demos File Schema Pad

Scott Ferretf has produced a short video demonstrating some of the advantages of the new File Schema Pad in Clarion 7.
Posted Thursday, October 23, 2008

DMC 1.5.0.3

DMC version 1.5.0.3 is now available. Changes include: During transfer of data added in column mapping options in operations the possibility to extract a portion of a column only (LEFT - RIGHT and SUBSTRING); Fixed bug in Portable Version where creation and synching were not working properly.
Posted Thursday, October 23, 2008

J-Html 2.01 For C7 AppGen

StrategyOnline has released J-Html 2.01, its first 100% Clarion 7 AppGen-ready product.
Posted Thursday, October 23, 2008

StrategyOnline Price Increases Postponed To Nov 4

StrategyOnline has postponed its price increases until November 4, 2008.
Posted Thursday, October 23, 2008

Clarion Third Party Profile Exchange October 21 2008 Update

The Clarion Third Party Profile Exchange has been updated. This is a major release of both online and data versions.
Posted Thursday, October 23, 2008

Clarion Folk Ad Post Summary

Stu has posted a summary of thoughts contributed to the "$10 Million" post/thread a few days ago.
Posted Thursday, October 23, 2008

SetupBuilder 6.9 Build 2390

SetupBuilder 6.9 Build 2390 is now available. This release provides support for the latest Microsoft OEM Ready program. This feature has been back ported from SetupBuilder 7. This release is available, free of charge, to all SetupBuilder customers who have an active SetupBuilder maintenance subscription plan.
Posted Thursday, October 23, 2008

DMC Birthday Special 25% Off

DMC is 1 year old, and until the end of October 2008 the price has been reduced by 25%. Save $100 on the Enterprise version.
Posted Friday, October 17, 2008

DMC 1.5.0.2

DMC 1.5.0.2 is now available. Changes include: Support for SQL Anywhere cloning; Real Edit In Place in the Viewer (with a calendar popup for all dates); Format a LONG to a DATE or a TIME column and export the data as is directly; After doing a QBE on any Table you can now delete all the records of the resulting subset; Embedded BLOB Viewer with predefined types for images; Complete log file creation; eMailing client to automatically send the logs in case of a crash; In the Mapping Options added the possibility to perform an operation on two columns (+ - * /); SQL Cloning button to rename the table (instead of the old entry field); SQL Cloning button for search and replace; Various enhancements.
Posted Friday, October 17, 2008

HCUG Meeting

The next HCUG meeting is at the Venu Map Trustco office, Unit 304 Oakmont Building, Somerset Links Office Park,(Off De Beers Ave) on October 22, 2008 (Wednesday) from 17:30 for 18:00. Duration is around 2-3 hours. Bring a cool drink or beer of your preference. Cover charge is R30 per person cash (for grub, coffee, tea etc). Topics: Continue from where we left off last; Q&A; Some surprise topics. Please R.S.V.P. to rsvp at clarionusergroups.com
Posted Friday, October 17, 2008

CapeSoft "C7 Is Coming" 20% Off Special

CapeSoft is having a "Clarion 7 is coming" special until October 19. Clarion 7 with the new AppGen is imminent, and CapeSoft is starting the celebrations with a 20% sale on all of its accessories, including the accessory bundles. CapeSoft

is also offering a 100% money back, no-quibble guarantee for 60 days (normally 31 days) after your purchase.
Posted Friday, October 17, 2008

What would you do?

Stu Andrews wants to know how you'd advertise Clarion if you had a ten million dollar budget.
Posted Friday, October 17, 2008

Clarion Desktop 4.20

Clarion Desktop 4.20 is available for download at no charge. This version includes a new, improved custom search feature (now searching 50 Clarion websites with a single click).
Posted Friday, October 17, 2008

BFlat 1.0

The BFlat templates from Comsoft7 are designed to give your apps a consistent look and feel. Supports C6 ABC and Legacy, C7 in Ver 1.0. Other versions will follow. Features include: Set system SDI; Set windows to NoFrame; Disable Esc key; Replace windows caption bar with BoCaption (and lose the red X); Allow window to be moved (dragged) with mouse in BoCaption; Use icons in BoCaption (16 x 16) full size for better look; "Small BoCaption" or panel options for BoCaption; BoCaption Panel Bevel +3 to - 3 for inner and outer bevels; BoCaption color fill; BoCaption font name; BoCaption font size (will cut off if gets too large); BoCaption font style; BoCaption font color; Font selection button in C6.x on; Import, export settings to INI file for multi-DLL use; Reset all procedures to new global settings button; Individual procedure control over BoCaption features.
Posted Friday, October 17, 2008

vuMail 2.0

Valutilities has released vuMail version 2.0. Major improvements include: Multi-threading (sends emails in the background); Sending consecutive emails from a file (CSV format); A freely re-distributable email setup and test program (provided in both executable and source code). vuMail is still available for $49.00 through October 17th, 2008. After that, the price goes up to $99.00.
Posted Friday, October 17, 2008

Noyantis CalendarPro Wrapper Template 1.09

Version 1.09 Beta of the Noyantis CalendarPro wrapper template has been released. Modifications include: Event reminder facility added; Additional colour themes added; Compiled in icons can now be specified for control images; Multilingual compatibility features added (see global extension); OCXDirectCommand and OCXDirectValue methods added; Additional embed points added; ShowDay method added; Bug fix to Start of Week. The new version can be downloaded from the Members area using the original download and registration details contained in your sales emails. Demo available.
Posted Friday, October 17, 2008

Combit 20% Off

Combit has announced a 20% discount on new subscriptions for version 14 of List & Label. Release is anticipated in Oct/Nov 2008. As a Subscriber - Professional or Enterprise Edition 13 - version 14 is already a part of your subscription package and so everything is dealt with and paid for. You will automatically receive information regarding download or CD delivery. Please register your software, if not already done. As a customer of the Standard Edition 13 or any edition 12

or older, you can get an early bird upgrade to Standard Edition 14 as long as you place your order before Oct 22, 2008. Save 20% on the regular upgrade or new license price. Please register your software in advance. Independent of edition and version, early bird subscriptions - Professional or Enterprise Edition 14 - are available to all List & Label customers until Oct 22, 2008. Pre-order and save 20% on the regular new subscription price.

Posted Friday, October 17, 2008

Clarion Third Party Profile Exchange Web Update Release

A web update to the Clarion Third Party Profile Exchange is now available.

Posted Friday, October 17, 2008

Clarion Magazine

First Look: AppGen

by Dave Harms

Published 2008-10-22

Yesterday SoftVelocity released the new Clarion 7 Application Generator to third party vendors for compatibility testing. Naturally, the third party vendors were very happy; a few individuals outside that group were disappointed not to be included. But I think it was a good choice to let the vendors beat on the new AppGen first, not just to ensure their templates are ready for the CSP beta but also to flush out the first round of bugs.

And there are some bugs, to be sure. I've managed to crash C7 a few times already, as have others. But, like the first IDE release, this is a remarkably stable product, and I don't think it will be long before it's released to the rest of the beta program participants.

Why should I care?

I've heard a few developers say they don't really care about C7 because it doesn't offer them anything they can't get in C6. This is true in the same way that a Bugatti Veyron offers no significant advantage over a Chrysler Minivan, inasmuch as they both have four wheels and an engine.

The problem is that there isn't one absolute killer feature in C7. Rather, there are a bunch of small to medium-size features that, put together, make for an enormous gain in productivity and ease of use. But it's hard to grasp the full significance of the sum of these features until you spend a little time actually using C7.

Goodies in C7 include, but are not limited to:

- A real editor with actual undo, regex search, etc.
- Code folding
- Color syntax in the embeditor
- A flatter user interface (not as flat as I had hoped, but very much better than it was, as I'll illustrate later)
- RTL improvements, including ClearType and Unicode support, styled menus, tabbed MDI windows
- The ability to generate and compile using earlier versions of Clarion (that is, the APP/DCT get migrated to C7 format, but in all other respects you have an app that is C5.5 or whatever version (back to C4 I think).
- IDE is no longer 16 bit so will run on Vista 64
- Multi-application solutions
- A much richer and highly customizable IDE
- The file schema pad
- Alignment helpers in the window designer
- Significantly faster code generation and compilation (for the SCHOOL app gen/compile time dropped by about 40%)

No doubt there's a bunch of other stuff I can't recall at the moment. I won't touch on all these points in this article; rather I'll try to give you an overall impression of the new AppGen.

New skin, same bones

In many ways the C7 AppGen looks much like the C6 AppGen. Figure 1 shows the C6 IDE with the SCHOOL app

open; Figure 2 shows the same app in C7, using the default IDE layout. Figure 3 is a closeup of the AppGen pad, squished a little to fit on the page without resizing.

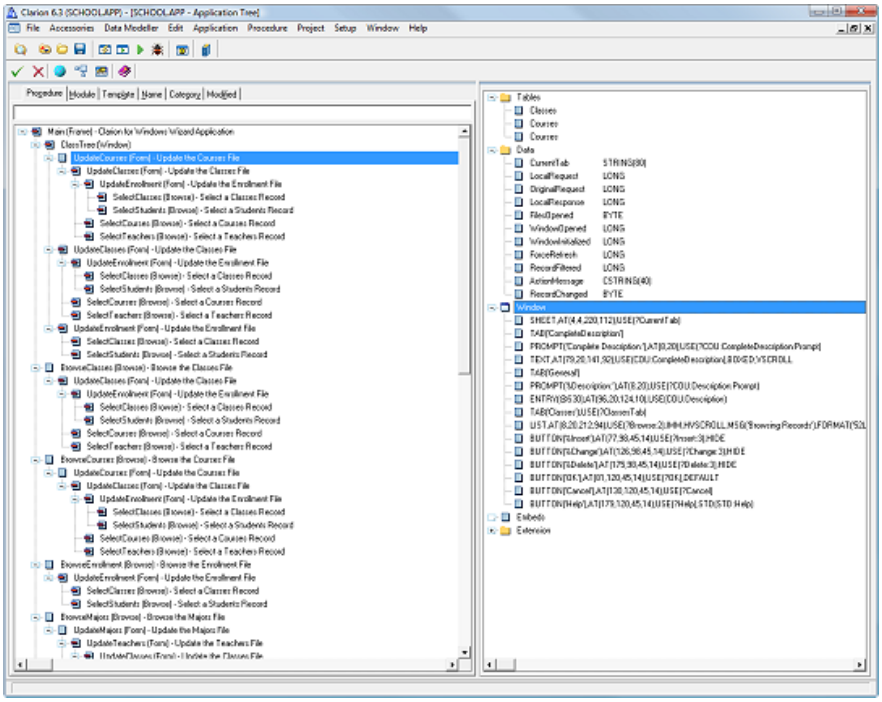


Figure 1. SCHOOL.APP in C6 (view full size image**)**

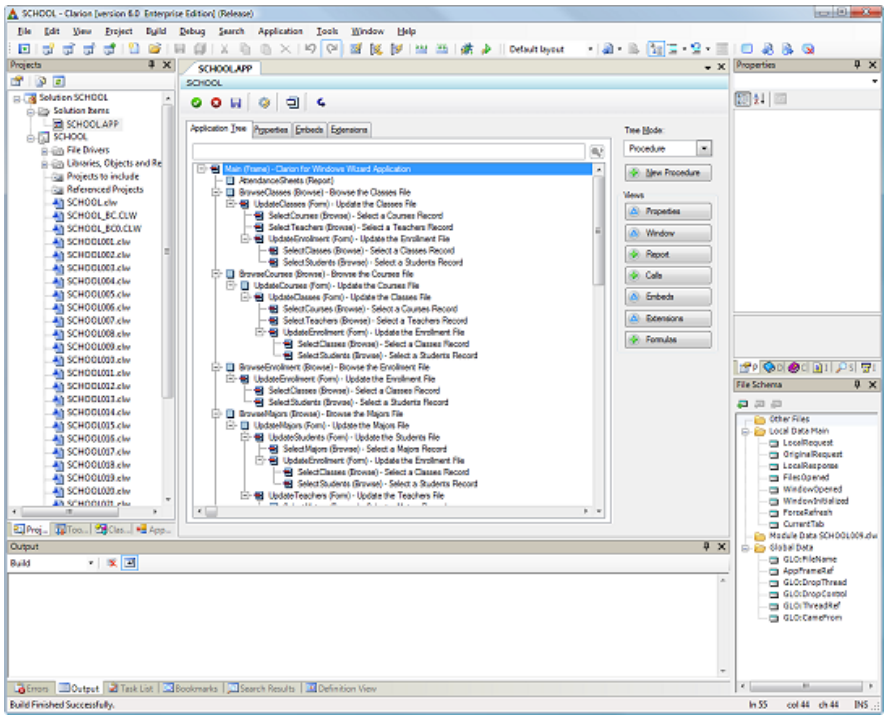


Figure 2. SCHOOL.APP in C7 (view full size image**)**

In Figure 2, take note of the various tabbed areas. On the left side you have the Project pad, but this area (in the default layout) is also used to display the Toolbox (which is active when you're in a window or report designer), the Class pad (handy for navigating), and the App pad (like the Project pad but showing only APP files). On the right side you have the Properties pad, which is active at various times, including when you're in a window or report designer, and below that

the File Schema pad, which lists files and variables for the selected procedure. At the bottom of the IDE you have a variety of informational pads, including Errors, Output, Task List, Bookmarks, and Search Results. I have no idea what Definition View is for.

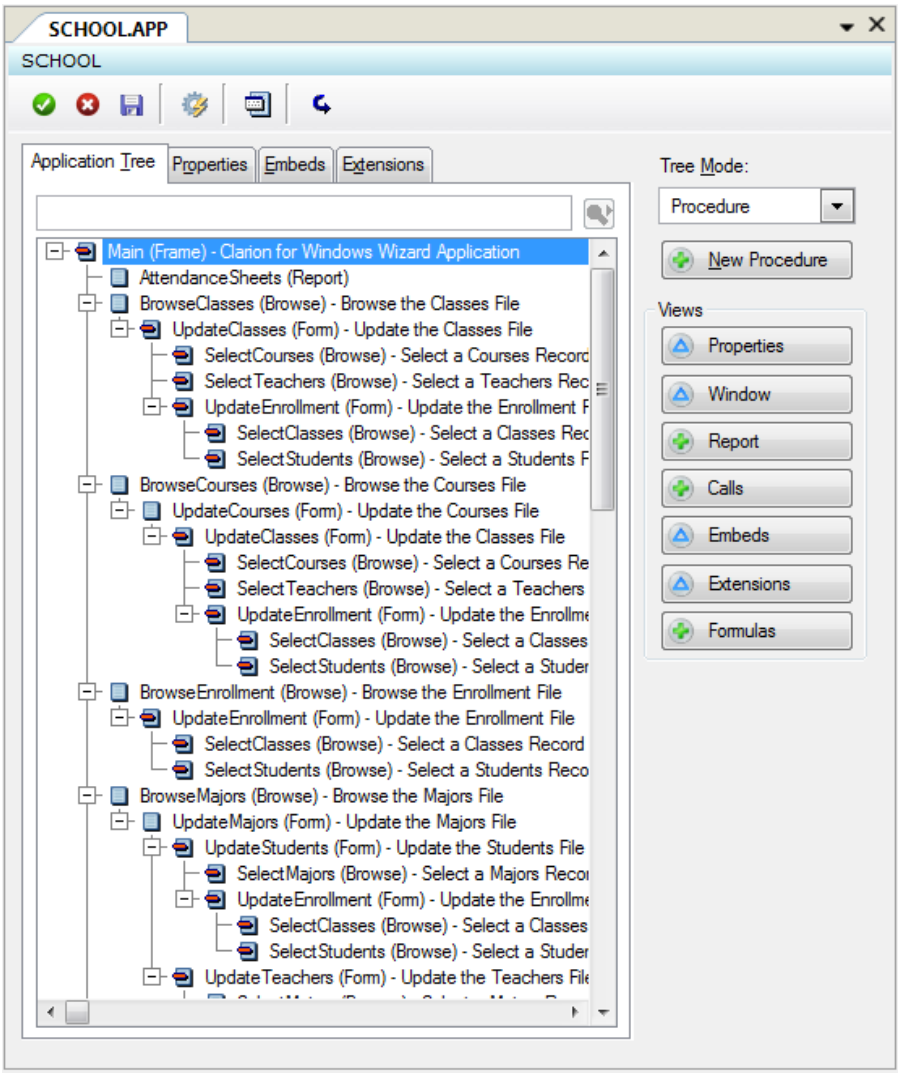


Figure 3. SCHOOL..APP in C7 - AppGen pad only

In Figure 3 note the tabs across the top, which provide access to global app settings, and the buttons down the right which for the most part correspond to the buttons in the procedure properties window in C6. This is the first click saved, as you can go directly to, say, the window or embeds without having to go to procedure properties first.

If you click on one of these buttons, such as Embeds, you'll see the window shown in Figure 4. Note that each of the buttons in Figure 3, with the exception of Formulas, is now represented by a tab. The tabs make it easy to navigate to the different areas with a single click, as compared to having to drill down, then save, then drill down again.

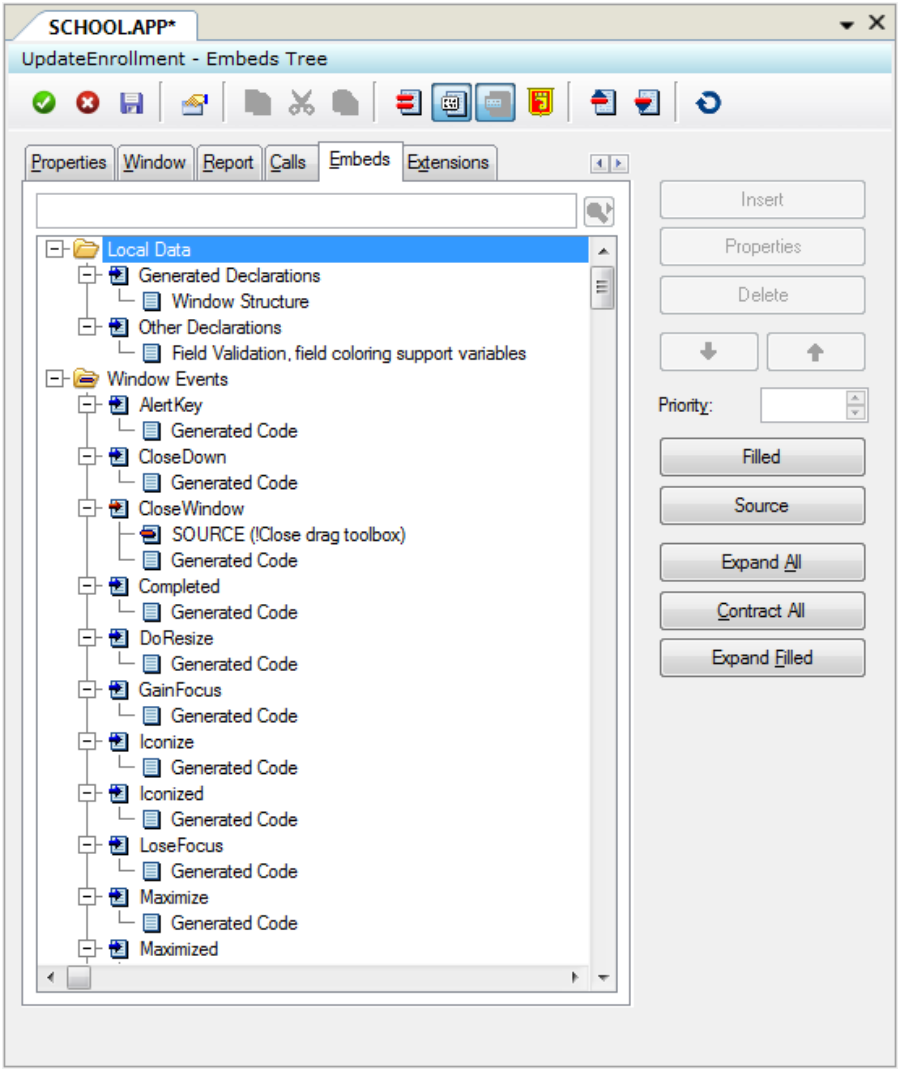


Figure 4. Procedure details.

There's one oddity here, however. If you click on the Window tab you'll see Figure 5. The default is to show just the window text. You'll need to click on the Designer button to see the window designer (Figure 6).

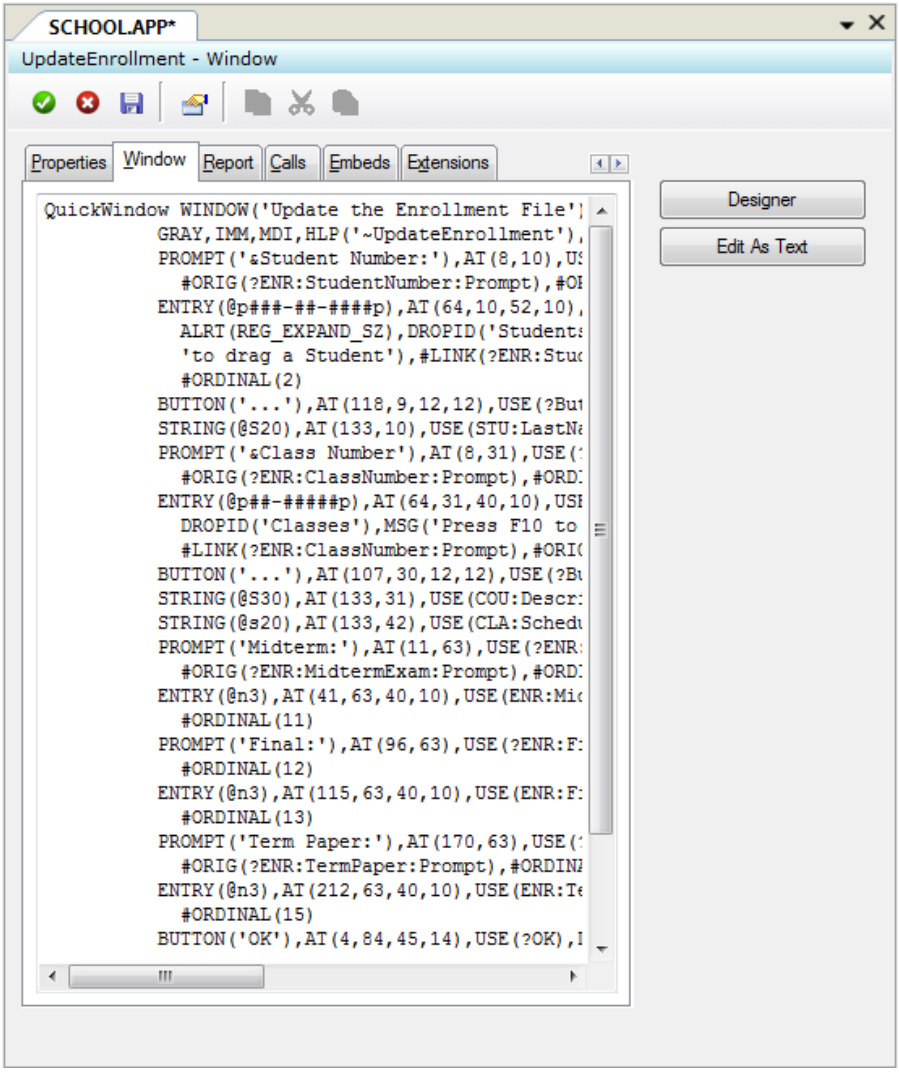


Figure 5. The Window tab.

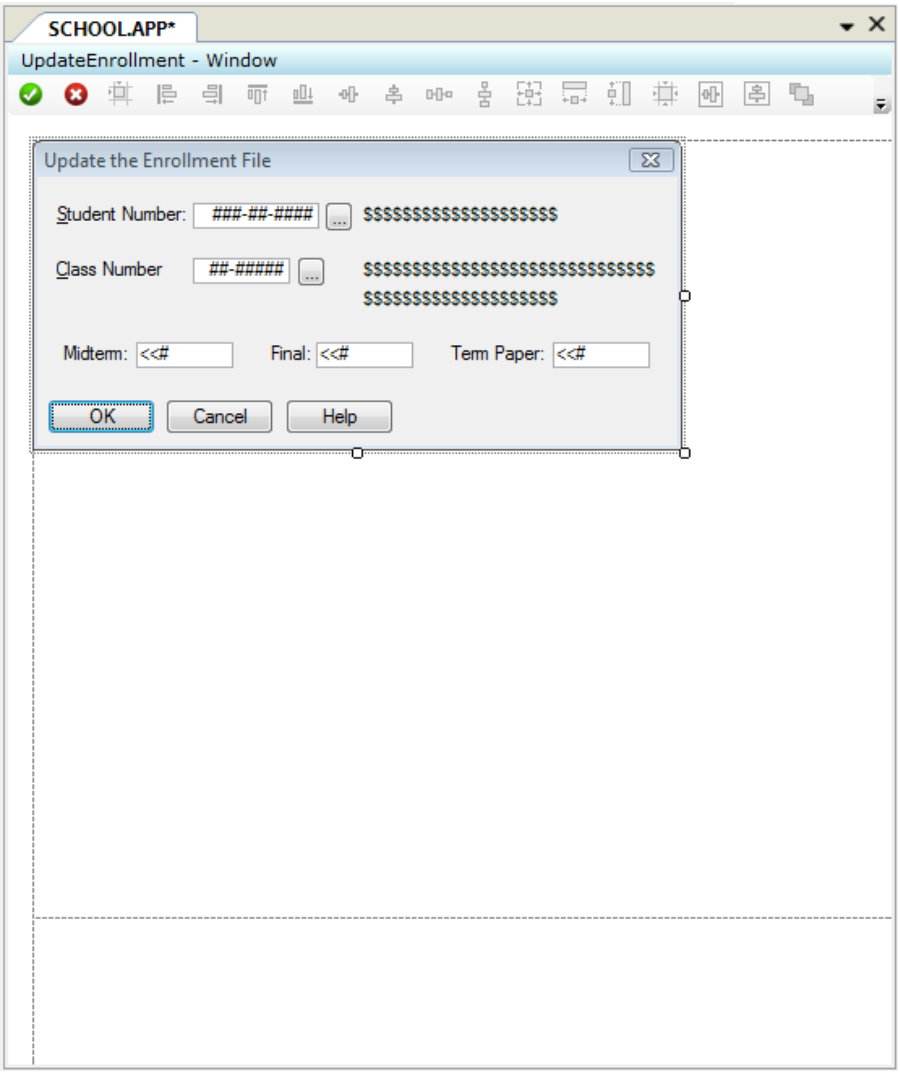


Figure 6. The window designer

You can set the designer view as the default, but if you do whenever you click on the Window tab you'll need to exit the window designer to regain access to your tabs.

While I'm in the window designer I should point out the new alignment helpers. Figure 7 shows the window formatter displaying an update form. I've just grabbed the entry field under the cursor and I've moved it slightly. As I move the entry field (or any other control) around the window, the formatter displays alignment lines. These lines indicate when any side of the control I'm moving aligns with any side of any other control. In this case the entry field aligns with the top of the prompt to the left and the string to the right, and with the left edge of the entry control below.

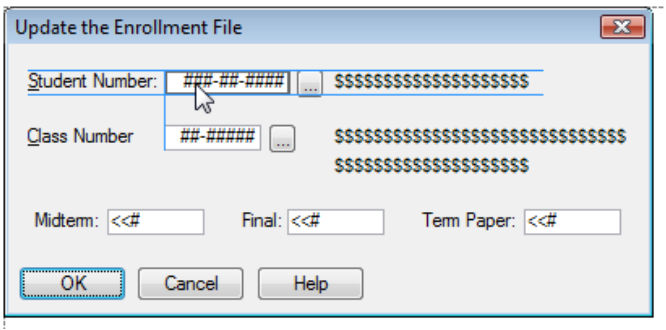


Figure 7. Moving controls in the window formatter.

The designer also snaps controls to these alignment lines, making it dead easy to put the controls where you want them.

There's another kind of snap line for spacing between controls. Figure 8 shows a prompt moved close to the left side of the window and the prompt just above. In both cases a short line perpendicular to the side of the control appears indicating the minimum distance has been reached.

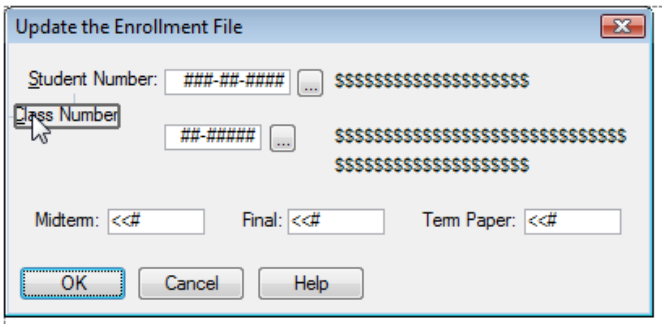
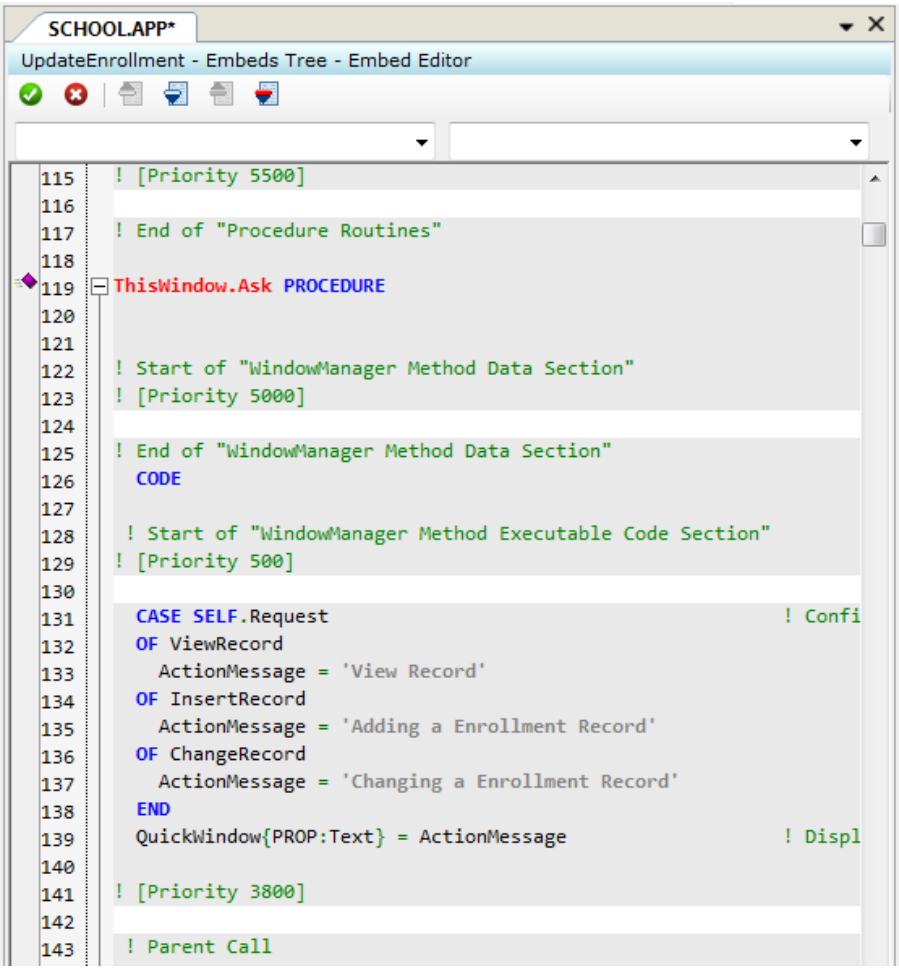


Figure 8. Snapping to minimum distances between controls

You can now drag and drop fields/variables, controls and control templates onto the window; in C6 you click once (or more) to select and click again to place.

Embed points

Embed points work as you'd expect, but with one very important bonus. The procedure-wide embed editor, PWEE (a.k.a. the embeditor), now features color syntax (Figure 9).



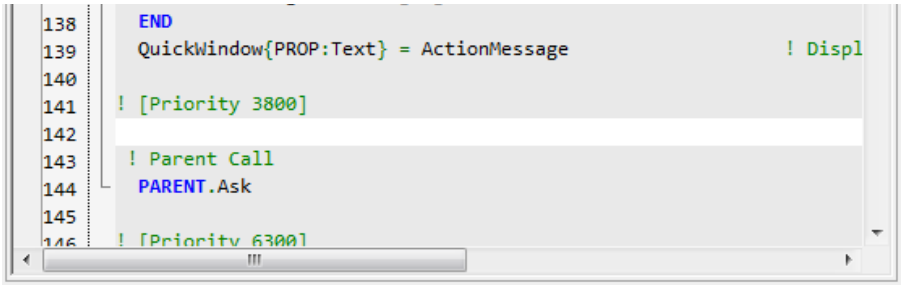


Figure 9. Color syntax in the embeditor.

If you've ever done work in the embeditor before you've probably suffered from eyestrain trying to differentiate embed comments from code. Color syntax takes care of the problem completely. As an aside, it's quite easy to not only specify the colors you want in C7 but to define your own syntax coloring rules. For instance, I've set up a few simple rules that make template writing a joy, so far as template writing can ever be that.

As Figure 9 also indicates, the embeditor includes code folding, but at present settings are not preserved between sessions. With straight source code the editor remembers code folding as returns you to your previous location the next time you open the file. It would be nice to see this done for the embeditor as well.

There's a great deal more I could say about C7 (and will say in the coming weeks and months) but for now I just want to touch on two more points: multiple Clarion versions and the flatness of the IDE.

Multiple Clarion versions

In order to open a C6 (or earlier, I assume) app in C7, that app (and any dictionary) must be converted to the C7 format. But *conversion* to C7 isn't the same thing as *migration* to C7.

Conversion simply means the APP (and if present, the DCT) is converted to the C7 file format, after which it cannot be read by a previous version of Clarion.

Migration means generating that application with C7 templates and compiling with the C7 compiler.

One of the truly wonderful features of the new IDE is that after converting the APP and DCT to C7 you can, if you wish, maintain that application using the compiler, templates, source files, libraries and DLLs *from a previous version of Clarion*. You can even switch back and forth. I've done with several times with the SCHOOL application, first generating and compiling it as a C7 application, then as a C6 application, and once again as a C7 application again.

C7's installation program detects eligible existing versions of Clarion and installs them in the IDE. To apply a version of Clarion you choose Build | Set Clarion version, and choose the version of Clarion you want. You will need to register the templates for that version of Clarion if you haven't already done so. Then open the APP and you're in business.

IDE flatness

In the early days following the initial C7 announcement much was made of the flatness we could expect in the new IDE. I'm not sure that vision has been fully realized, but there's definitely less mouse clicking involved. Just as importantly, the new IDE is less modal than the old IDE.

Case in point: I loaded up the HowTo-ABC browse sample application, and encountered one compile error due to an empty ICON attribute. In C6, and in C7, an error in a window structure won't take you to an embed point; instead, you'll get the option to go see the offending generated source code. In C6 that window is a modal window. You have to exit to get back to the application tree to fix the problem, by which time (if you're like me) you may have forgotten the name of the procedure and the offending line of code. In C7 I similarly loaded up the source containing the error, but left the source window open for reference while I navigated to the procedure in question. I was still able to switch between the AppGen window and the source window when I had the window designer open. That kind of flexibility is most welcome.

Summary

In short, the new AppGen works. Not all the time, of course - this is a beta after all. But it's also quite possible to have a session whereby you open an application, make changes, generate the code, compile and run and have everything work perfectly. AppGen is well on its way.

There's a great deal about the AppGen I haven't touched on, and much of that has to do with IDE enhancements such as the Property Pad and the File Schema Pad. As well, the ability to have solutions containing multiple APPs (as in multi-DLL applications) is of great value to many developers. Beyond AppGen, improvements to the RTL make it possible to deliver a more sophisticated and visually pleasing user interface. Unicode support is finally here, and at last we have an IDE that runs on 64 bit Windows. (And as Steve Parker pointed out to me, because of C7's support for multiple Clarion versions you can go ahead and install your older Clarion versions on Vista 64 and make use of them even though the IDEs themselves won't run.)

Stay tuned...

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).

Reader Comments

Posted on Thursday, October 23, 2008 by Stephen Ryan

Well done SV. Use the new Clarion sharp on all new projects.

Posted on Thursday, October 23, 2008 by douglas johnson

David,

Might you touch on how a .red file is handled? If multiple apps are open, is only one .red file in effect?

Posted on Thursday, October 23, 2008 by Dave Harms

Douglas,

Redirection files are handled pretty much the same way as in previous versions (although there is now an INCLUDE directive).

AFAIK you can't have two apps in one solution that use different Clarion versions, so if you've set the version to, say, C6 then the C6 RED applies to all the apps in the solution. And of course you can still have local RED files in the app directories.

Dave

Posted on Thursday, October 23, 2008 by Scott Ferrett

The redirection system works on a version and directory basis. So if you are building a solution using Clarion 6 and that solution consists of 2 apps in different directories, then you can have a C60EE.RED file in each directory. The redirection file in effect depends on which app you have active.

One feature of C7 is the INCLUDE directive. With this you can include other redirection files. Note that you can use this feature of the c7 redirection system even when working with older versions of clarion (although you will have to enable this via the Tools/Options/Clarion/Versions screen for versions before C7. Another thing you can do is change the name of the redirection file. So if you change the name of the redirection file for C6 to C60FromC7.red and enable include you can then have a a C7 system that does not clash with your C6 system but has the power of the C7 system.

Posted on Thursday, October 23, 2008 by Philip Prohm

In Figure 3, Tree Mode is a droplist. In the 16-bit IDE, changing the View in the Application Tree Dialog is move-click (or Ctrl+Tab). However, in the 32-bit IDE, changing the Tree Mode is move-click-move-click, a step backward.

Can the droplist please be changed to a listbox (reinstating move-click)? There's room for a listbox. Thanks. Ctrl+Tab would also be good if it's not there already.

While I'm here: I hope maximum keyboard functionality is maintained. While a mouse has its moments, when you have two hands on the keyboard you want to be able to do things via key-combinations not take a hand off the keyboard just to use a mouse.

Lack of 32-bit IDE was my biggest gripe so goodonyer SV for fixing this shortcoming.

Posted on Friday, October 24, 2008 by Bruce Johnson

Hi Philip,

There is a button on the app's toolbar, the last one on the right, which "toggles" between the last 2 modes you used. So for example, I find myself oscillating mostly between say Module View, and Procedure view, then this button can do that.

Or if I'm oscillating between Date-Modified view and Procedure view it does that too.

Perhaps not as east as C6, but it's a bit quicker than only using the drop-down.

Cheers
Bruce

Posted on Friday, October 24, 2008 by Randy Rogers

Hi Dave
Thanks for the article. Yes lots of nice little things but I saw no mention of built-in, out-of-the-box support (like VB eg) for COM objects. Is this important bit still missing from Clarion7? Do I still need to write COM wrapper classes to use com objects?
Randy

Posted on Friday, October 24, 2008 by Dave Harms

Randy,

I don't use VB so I'm not sure what you're expecting, but COM support looks similar to C6 from what I can see. Perhaps someone with more COM experience can jump in here.

Dave

Posted on Friday, October 24, 2008 by Jane Fleming

Thanks, Dave :)

Question -
"C7's installation program detects eligible existing versions of Clarion and installs them in the IDE."
How does that detection work for those of us who are running C6 in VMWare in Vista 64 but want to run the new IDE/Appgen natively?
Is it just a matter of installing C6 in the host machine and not attempting to run the C6 IDE?

Posted on Friday, October 24, 2008 by Dave Harms

Jane,

Pretty much, although I did find with 6.3 that I couldn't register the templates right away. I copied across my working C6 folder from a 32 bit machine and all was well (although perhaps just restarting the C7 IDE would have accomplished the same thing).

Dave

Posted on Saturday, October 25, 2008 by Rakesh Khatri

Dave -

Are there template enhancements to handle Classes created in App to be exported. Currently in a multi dll app, class declarations are not exported. You have to manually created export list etc. Any change in this area?

Rakesh

Posted on Monday, October 27, 2008 by Dave Harms

Rakesh,

I'm not aware of any enhancements to the templates regarding class exports as of the current beta.

Re exporting classes you may want to have a look at Ben Dell's article at http://www.clarionmag.com/cmag/v10/v10n06classdll.html

Dave

Add a comment

Clarion Magazine

AppGen Beta Status Report

by Dave Harms

Published 2008-10-31

Third party vendors have had their grubby mitts on the first AppGen release for a little over a week now. And although the stated purpose of this release is to ensure compatibility with third party templates, a number of bug reports have been filed for other areas as well.

First, the template issues.

Templates compatibility

The C7 template parser enforces the template language syntax more strictly than the C6 parser. One of the first places I noticed this was in #DEFAULT sections in templates. Among other things, #DEFAULTs let you declare windows, reports and controls using familiar structures. And C6, as it turns out, isn't picky about whether or not these structures have closing END statements, which results in one of the most common errors.

In C6 templates the following is legal:

```
CONTROLS
  BUTTON('...'),AT(,,12,12),USE(?LookupFile)
```

In C7 you need to add that closing END:

```
CONTROLS
  BUTTON('...'),AT(,,12,12),USE(?LookupFile)
END
```

Other template issues you could previously get away with include omitted parameters to SLICE, duplicated attributes, missing commas between attributes and the like. And apparently C6 will let you use #TABs without an enclosing #SHEET! Who knew?

Lee White, who pushes the template language farther than anyone I know, has found some problems with an esoteric use of #FIND.

On the whole the template problems discovered so far have more to do with improper usage than bugs, although several parser bugs have been found and reportedly fixed.

So far it seems that any fixes to the templates are backward compatible - that is, if you fix up a C6 (or earlier) template so it can be registered in C7, you can still register that template in the earlier version of Clarion.

And for anyone who missed it, Clarion supports compiling applications for any release of Clarion you have on your system, going back to CW1.5 (which is the first version with a 32 bit compiler). Your apps must be converted to the C7 APP and DCT file formats, but otherwise you can work with, say, C5.5 apps if you're constrained to do so. Interestingly, this suggests that you can also keep multiple versions of C7 on hand (you'll have to make copies of the relevant directories and add the version manually) which would make upgrading to a newer version a much less risky proposition (provided, of course, that the showstoppers aren't in the IDE itself...).

There is one important bug in multi-version support right now: converting applications is slightly broken. If you specify an earlier Clarion version, then open an app file, the app is always converted to the C7 app file format, as it should, but the specified build gets reset to the current C7 build so the C7 templates are applied to the app. You have to close the app, specify the correct version and open the app again. That's not a big deal for C6 apps since the template chains are almost identical, but if you try to do this with, say, a CW20 app, the C7 templates will add a bunch of stuff to the app that's incompatible with CW20. This is a reported, confirmed bug and I hope it will be fixed for the next release.

Compiler/RTL issues

There don't seem to be any compiler issues, at least none that I can recall. That's not a surprise since the hand-coders version has been out for some time now and the compiler has had a decent workout.

As for the RTL, there have been a few complaints about some of the new visuals, in particular consistency of menu appearance.

COM

A number of developers have asked whether COM support is improved in C7. I've been told yes, it's improved, but the improvements appear to be mainly in the RTL rather than in the IDE. That is, there's no great difference in how COM objects are handled in the window designer, at least that I can see, except that the list of ActiveX objects is now populated.

AppGen bugs

There have been a variety of AppGen bugs reported, some minor, some crashworthy. I'm not seeing any one part of the AppGen as having more issues than another. Problems include display problems after resizing, message dialogs hiding other message dialogs, occasional exceptions (usually allowing you to continue working), a few menu items not working and the like. There has been some discussion about potential memory leaks, with a few developers convinced they are happening, and SV certainly wanting to know of that's the case but not being able to confirm any major leaks. I've posted a PTSS on this as well, showing that opening the DLLTutor solution, generating all, closing, and repeating the entire process results in continually higher memory usage in Task Manager. Bob Z has pointed out that due to the nature of .NET memory management Task Manager doesn't always report actual memory usage with accuracy.

One thing is certain: C7 makes it possible to use more memory than ever before. In C6 you can only load one application at once; in C7 you can load, in theory, hundreds. And if you want to do that you're going to use up a lot of memory. Although 32 bit Windows lets you install 4GB of RAM, the max usable amount is more like 3GB. For really large multi-app solutions I don't think a 32 bit version of Windows is going to be sufficient. Start thinking about Vista 64, which lets you address up to 128GB of RAM. (If Vista scares you, keep in mind that a number of Clarion developers have reported that Vista 64 is in fact much more stable than Vista 32.) And maybe it's time for a new processor as well. Yes, C7 generates and compiles code much faster than C6, but multi-app support means way more potential for pushing a development machine to its limits.

> An aside on multi-core processors: as Bruce Johnson pointed out in the newsgroups, multi-core processor ratings may lead you to think that a dual core chip is twice as fast as a single core chip. In fact the dual core chip may run any given single-threaded task slower than the comparable single core chip due to the multi-core management overhead. And tasks like compiling are single-threaded, as far as I know. Now if we could get multi-threaded code generation (and I can't think of a reason why not), multi-core chips could present a significant advantage....

For those of us who are keyboard-intensive in our use of AppGen, the new IDE is irritatingly mouse-intensive. I'm hopeful that we'll get more keyboard shortcuts eventually. As it is, menu items are still subject to change and it makes sense not to add too many accelerator keys until the IDE has settled down. Related to that, I've noticed that I can click on the "generate, compile and run" (GC&R) button from anywhere in the IDE, even the embeditor. That doesn't save anything I've been working on, but if it would that would be way cool. Imagine being able to stay in the embeditor (or the window designer, or whatever) and compile and run your code with a single click. I wouldn't care so much about my

missing keyboard shortcuts. If that approach isn't implemented then I think we'll need to see the GC&R button disabled, and some of us at least will absolutely have to have those keyboard shortcuts for saving work and closing windows.

One rather important item not yet fixed is that compile errors in embedded code take you to the generated source, not the embeditor. In C6 that kind of problem would make the IDE unusable; in C7 you can split the AppGen pad by clicking on a source file tab and dragging it to the right side of the pad. This way you can view the generated source (with the error) along side the embed list or even the embeditor. That's pretty cool.

The numbers

As of yesterday (Oct 30) there are a little over 70 public (to beta participants, at least) bug reports posted since the start of the AppGen beta. Most of these relate to the templates or the AppGen itself. Of these 18 are listed as confirmed, not fixed, and five are opened for review (meaning they're not yet confirmed as bugs). Another 26 are listed as closed (which can mean fixed, or expected behavior, or unable to reproduce). That leaves about 20 reports in some other state, usually meaning additional comments have been posted, or further information has been requested from the developer.

Several of the reports I've filed were for bugs that were already fixed in SV's internal builds. The first AppGen release is build 4240; the highest build in the PTSS is already 4382.

What to expect

I can't tell you how long the third party test phase will last, but I'm hoping only a few more weeks. I do expect (and this is *only* a guess) SV will release at least one more build to third party vendors before opening AppGen up to the rest of the CSP, just to allow for another round of compatibility testing (particularly since there are fixes that will affect at least some third party templates).

I further expect that once the C7 beta goes out to CSP participants it will be usable by most of those developers. There will still be bugs, and some of those may even be showstoppers for a small minority. On the other hand, SV has the once-in-blue-moon luxury of working with a completely rewritten code base, using a managed code environment that makes it far easier to track down bugs than when writing unmanaged code.

As with the initial hand-coder beta releases of C7 and Clarion#, I expect to see fairly quick progress on reported bugs. The only thing I think really should've been there by beta is compiler errors in embed code being returned to the embeditor rather than to source. All the same I wouldn't have wanted SV to hold back the beta a minute longer.

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).

Reader Comments

Add a comment

Clarion Magazine

Understanding The C7 Build System

by Dave Harms

Published 2008-10-30

One of SoftVelocity's stated goals with Clarion 7 was to make the process of developing programs in AppGen familar to Clarion 6 users. And the process of creating applications in C7 is very much like the C6 process: although there are some new features and changed windows, much of what you see in the C7 AppGen looks familiar.

To be more specific, the process of creating the application is fairly familiar. But that's only part of the story. After you create the application (and generate the source code) you need to compile and link it. And here C7 is radically different from C6.

Terminology

One of the difficulties in discussing the C7 compile/link process has to do with terminology, specifically with the use of the term "project." Most C6 developers, if they think of projects at all, think of hand coded applications. The simplest hand coded Clarion program (think "Hello World") consists of a single source file (ending in .CLW) and a single project file (ending in .PRJ or .PR). The PRJ (or PR) file is what the Clarion build system uses to determine what steps it needs to take to turn your source code into an EXE (or DLL or LIB).

Here's a Hello World program (hello.clw):

```
program

map
end



code
message('Hello world')
```

And here's a PRJ (hello.prj) to go with it:

```
--
#noedit
#system win32
#model clarion dll
#pragma debug(vid=>full)
#compile "hello.clw"
#link "hello.exe"
```

The PRJ file tells the Clarion build system some information about the kind of application it's to create, then specifies that hello.clw is to be compiled, and finally that the resulting binary code should be linked into an executable. The details of the process aren't important here - what matters is that there's some project (build) data present.

Although you may not realize it, app files also contain project data. Open an app and choose File | Export Project File.

(In earlier versions of Clarion you had to either export a TXA to get that information or, if you were brave, open up the app in a text editor, being careful to to save the file.) Here's the project data for SCHOOL.APP:

```
-- Generator
#noedit
#system win32
#model clarion dll
#set RELEASE = on
#pragma debug(vid=>off)
#pragma optimize(cpu=>386)
#pragma define(_ABCDllMode_=>0)
#pragma define(_ABCLinkMode_=>1)
#pragma link_option(icon=>"_SoftVUn.ico")
#compile "SCHOOL_BC0.CLW" -- GENERATED
#compile "SCHOOL_BC.CLW" -- GENERATED
#compile "School.clw" -- GENERATED
#compile "SCHOOL001.clw" -- GENERATED
#compile "SCHOOL002.clw" -- GENERATED
#compile "SCHOOL003.clw" -- GENERATED
#compile "SCHOOL004.clw" -- GENERATED
#pragma link("C%V%TPS%X%%L%.LIB") -- GENERATED
#pragma link("Open.ico") -- GENERATED
#pragma link("Closed.ico") -- GENERATED
#link "school.EXE"
```

There's a little more information here, including some pragma statements telling the linker to include several icons in the EXE.

All of the build information in C6 (and prior versions) is centered on the concept of a single build target. That is, you can make a LIB, or a DLL, or an EXE, but really there's no built-in way to create a bunch of EXEs or DLLs or LIBs or any combination thereof, all in one go. If you want to do something like build a multi-DLL application in a single step you need a third party batch compiler.

Enter C7.

Build information in C7

In C7 you can have build information for a single program, as in C6, or you can have build information for multiple programs, as when you want to build that multi-DLL app. C7's build system is a superset of Microsoft's MSBuild.

There are two key concepts in MSBuild that relate to the changes in how C7 build applications. First is the idea of a *solution*. A solution encompasses all the pieces you need to build a single project, whatever that product may be.

A multi-DLL application is a typical example of a solution. The individual parts of that solution (the EXE and the DLLs) are each, in MSBuild terms, *projects*. A solution will always have at least one project. That's a vital point which I'll come back to later.

The potential for confusion has to do with how MSBuild and Clarion each refer to projects. In Clarion terms, a project is a hand coded application, and an application is something you build with the AppGen. In MSBuild terms, both of those things are simply projects.

> **NOTE:** In the first beta AppGen release the IDE uses the MS terminology; for subsequent releases the IDE dialogs have been changed to reflect a more Clarion-like usage.

Converting PRJ data to XML

When you open a C6 (or earlier) application in C7 the IDE warns you that the app file will be converted to the C7 format. You'll get a similar warning on any unconverted dictionary used by that application. Figure 1 shows the Project pad displaying the converted app. The Project pad has been renamed to the Solution Explorer, so I'll use that term from here on.

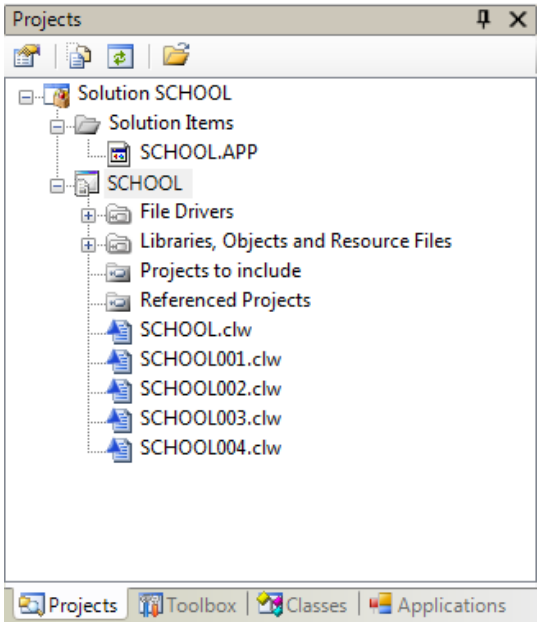


Figure 1. The Project pad (since renamed to the Solution Explorer)

Assuming you proceed with the conversion(s) you'll end up with two new files, a solution file ending in .sln and a project file ending in .cwproj. The top line in the Solution Explorer tree corresponds to the solution file.

> **NOTE:** You won't normally need (or want) to edit either the .sln or the .cwproj files directly. Instead you'll work directly in the Solution Explorer and let the IDE write out whatever changes are needed.

Here's the solution file for SCHOOL.APP:

```
Microsoft Visual Studio Solution File, Format Version 9.00
# Visual Studio 2005
# Clarion 2.1.0.2447
Project("{12B76EC0-1D7B-4FA7-A7D0-C524288B48A1}") = "SCHOOL", "SCHOOL.cwproj", "{1A130203-940F-4307-843A-F11FC9D87663}"
EndProject
Project("{2150E333-8FDC-42A3-9474-1A3956D46DE8}") = "Solution Items", "Solution Items", "{{2150E333-8FDC-42A3-9474-1A3956D46DE8}}"
    ProjectSection(SolutionItems) = postProject
        SCHOOL.APP = SCHOOL.APP
    EndProjectSection
EndProject
Global
    GlobalSection(SolutionConfigurationPlatforms) = preSolution
        Debug|Win32 = Debug|Win32
```

```
        Release|Win32 = Release|Win32
    EndGlobalSection
    GlobalSection(ProjectConfigurationPlatforms) = postSolution
        {1A130203-940F-4307-843A-F11FC9D87663}.Debug|Win32.Build.0 = Debug|Win32
        {1A130203-940F-4307-843A-F11FC9D87663}.Debug|Win32.ActiveCfg = Debug|Win32
        {1A130203-940F-4307-843A-F11FC9D87663}.Release|Win32.Build.0 = Release|Win32
        {1A130203-940F-4307-843A-F11FC9D87663}.Release|Win32.ActiveCfg = Release|Win32
    EndGlobalSection
EndGlobal
```

You'll see two project references in the solution file. One points to SCHOOL.cwproj, which is a compilable project, and the other simply appears to be a means of displaying details for the SCHOOL application, as in Figure 1. Keep in mind that this screen shot is from the first beta, and the display of app-related files may yet change.

The separation of the app-related info and the generated source code is an important one, as will become clearer a little later in this article.

While solution files are simply text files, .cwproj files are XML files. Here's the project file for SCHOOL.APP:

```
<Project DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <ProjectGuid>{1A130203-940F-4307-843A-F11FC9D87663}</ProjectGuid>
    <Configuration Condition=" '$(Configuration)' == " ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == " ">Win32</Platform>
    <OutputType>Exe</OutputType>
    <AssemblyName>SCHOOL</AssemblyName>
    <OutputName>school</OutputName>
    <ApplicationIcon>_SoftVUn.ico</ApplicationIcon>
    <DefineConstants>_ABCDllMode_=>0%3b_ABCLinkMode_=>1</DefineConstants>
    <Model>Dll</Model>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)' == 'Debug' ">
    <DebugSymbols>True</DebugSymbols>
    <DebugType>Full</DebugType>
    <vid>full</vid>
    <check_stack>True</check_stack>
    <check_index>True</check_index>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)' == 'Release' ">
    <DebugSymbols>False</DebugSymbols>
    <DebugType>None</DebugType>
    <vid>off</vid>
    <check_stack>False</check_stack>
    <check_index>False</check_index>
  </PropertyGroup>
  <ItemGroup>
    <FileDriver Include="TOPSPEED" />
    <Library Include="Closed.ico" />
```

```
    <Library Include="Open.ico" />
    <Compile Include="SCHOOL.clw">
      <Generated>True</Generated>
    </Compile>
    <Compile Include="SCHOOL001.clw">
      <Generated>True</Generated>
    </Compile>
    <Compile Include="SCHOOL002.clw">
      <Generated>True</Generated>
    </Compile>
    <Compile Include="SCHOOL003.clw">
      <Generated>True</Generated>
    </Compile>
    <Compile Include="SCHOOL004.clw">
      <Generated>True</Generated>
    </Compile>
    <Compile Include="SCHOOL_BC.CLW">
      <Generated>True</Generated>
    </Compile>
    <Compile Include="SCHOOL_BC0.CLW">
      <Generated>True</Generated>
    </Compile>
  </ItemGroup>
  <Import Project="$(ClarionBinPath)\SoftVelocity.Build.Clarion.targets" />
</Project>
```

The cwproj file contains all the information normally contained in a C6 app's project data. In fact, there's no longer an option to export project data from the app as this isn't necessary. (Actually the information still resides in the .app file - if you copy a C7 app to a new location, minus its .sln and cwproj files, and open that app, the IDE will simply create new .sln and .cwproj files.)

In previous versions of Clarion, building the application always means generating and compiling. In the first release of C7 with AppGen you can, if you want, build without generating the app. Simply press F8 to compile whatever source is there, or right-click on the Solution entry (the first item) in the Solution Explorer and choose Build. You can also right-click on any line corresponding to a .cwproj item (the SCHOOL line in Figure 1) and choose Build to just build that one application (only an advantage if you have more than one app in the solution, obviously).

There are also toolbar buttons, as shown in Figure 2. From left to right, they perform the following actions:

- Generate the currently selected application
- Generate all open applications
- Generate all applications
- Build project
- Clean and build project
- Generate code and start debugging
- Generate code and run the program without the debugger

Figure 2. Toolbar buttons

The ability to compile without generating is important for multi-app solutions. Figure 3 shows the Solution Explorer for the DLLTutor solution. Note that all I had to do here was open DLLTUTOR.APP; C7 loaded up all the referenced apps and created the multi-app solution (and yes it did all generate, compile and run).

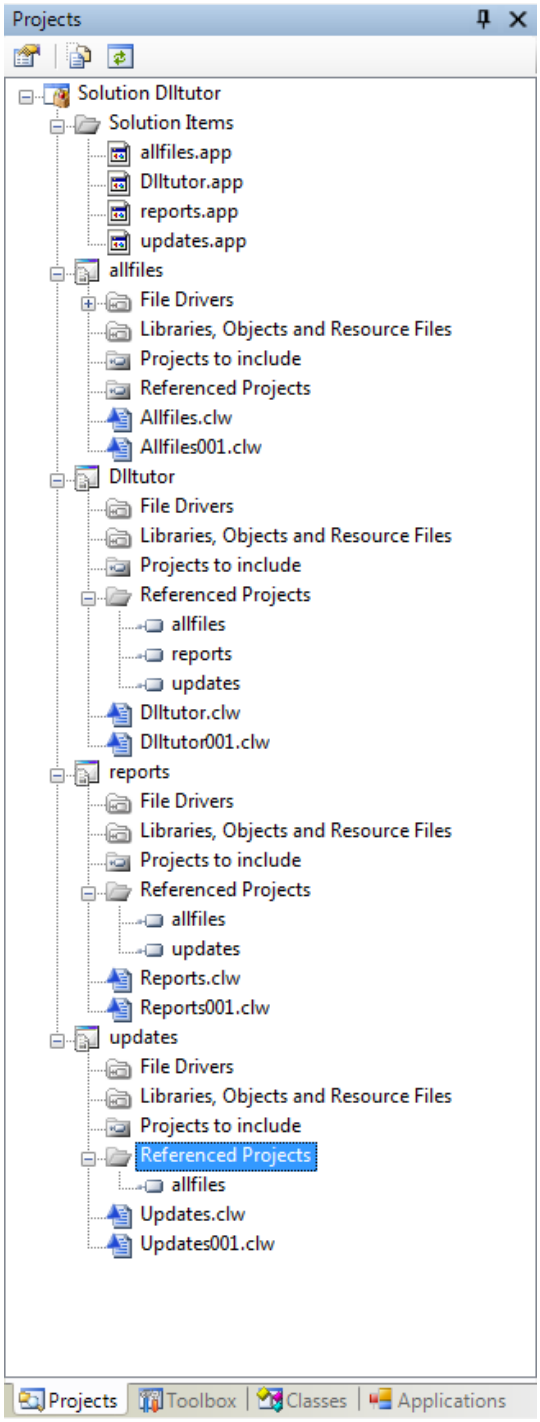


Figure 3. The DLLTutor application

If you have more than one application you probably only want to generate the code for the app you're specifically working on; you may also only want to compile that one app most of the time.

The build system compiles your apps based on dependencies. Note the Referenced Projects items in Figure 3. I can right-

click on AllFiles and choose Build and only AllFiles will be compiled as it has no referenced projects. If I build Updates, the build system looks at AllFiles to see if it's changed since the last build; if so, AllFiles is built first. (Actually that's not the case in the first AppGen beta - referenced projects, at least on my machine, are *always* built, but SV has confirmed that in their current builds this is fixed, and that's the behavior you can expect in the next beta.)

Loading sub-apps separately

The bigger the multi-DLL solution, the longer the load time and the more memory, so how about just loading individual apps? I tried this by opening the Updates app's .cwproj file. The IDE created a new solution file for that .cwproj, which was good. But no app showed up in the Solution Explorer. But that stands to reason, since the .cwproj contains only the generated source. I right-clicked on the solution and used Add | Existing Project to add the app file. Due to a bug (also reported) the app didn't show up in the Solution Explorer and the source project now showed up twice. I closed the solution and re-opened it and all was well.

At this point I could open the app, make changes and recompile. What's interesting about this approach is that when you open a single app with references to other apps, those referenced apps are never generated, at least in this build. And I hope it stays that way.

Remember that the solution file has two projects, one for the app and one for the generated source. Only the generated source has a corresponding .cwproj file.

The related projects are stored inside .cwproj files in a node that begins like this:

```
<ProjectReference Include="allfiles.cwproj">
```

That's right, you have nested .cwproj files. And since apps are in solutions and source is in projects, in this example when I build the Updates app I only generate source for Updates (since that's the only app in the solution); the source for AllFiles is simply compiled (if necessary).

In short, the behavior is exactly the same as if I'd built the app as part of the DllTutor solution, except that code generation can only happen for the one app in the solution. Presumably I could have a couple of apps in there if I wanted. This looks to me like a great way to work work a subset of a very large multi-DLL application. And of course you'd want to periodically load up the solution with all the apps in it and do a build to ensure all dependencies are resolved.

One more little trick

In any case, the ability is compile without generating is also useful for debugging. On more than one occasion I've found it helpful to hack away at generated code, compiling and testing until I've found the solution to some tricky problem. In the past I've done this by exporting the PRJ from the app, then using the PRJ. With C7 this step is no longer necessary.

Summary

There's a fair bit more to be said about the project system, but I'll leave that for another time, after the beta process has settled things down a little. The main thing to take away from this article is the new scheme of solutions and projects, and C7's ability to manage your multi-DLL applications all in one go.

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).

Reader Comments

Add a comment

Clarion Magazine

Trading Global Data for Data Obfuscation, Sort Of...

by Mark Geisinger

Published 2008-10-23

Anyone that frequents the Clarion newsgroups will recall the recurring questions and discussion of global data. Opinions vary from "Who cares?" to "It slows your program and is to be avoided." The reality is a bit more boring.

Global data is a good thing. You wouldn't get far without it. Like any data you create, it should be used when it is the best fit for the need at hand and with thoughtful consideration of its implementation and access methods.

Two aspects of global data use I don't recall being discussed are minimizing the potential for inducing errors and easing the effort required to maintain globals across multiple DLLs. A bit of defensive programming is always good, even with simple variable data access. By the end of this article I hope you will agree that I have reduced the error potential of my target case to its reasonable minimum. The second aspect of global data use, code maintenance, is the primary reason I created the cosOptions Class.

A Case for Globals

I have a project with nearly 100 global variables. Roughly one third are my own, with the remainder belonging to a third-party product. These globals are declared in four purpose-specific groups, three of which are for my own variables: user options, system options (for multi-user environments) and licensing. I also moved the third-party data into a group.

I'll illustrate the use of the cosOptions class with the user options group. In this multi-DLL project the gUserOptions group is declared as an instance of a TYPE and exported from the data DLL. It is used in code that requires early execution within the data DLL and is also used in every other module in the project.

Here is a subset of the gUserOptions TYPEd GROUP:

```
GRP_USER_OPTIONS_XML group, type
szBrwsScrollCtrlType       cstring(21)
szShowRecordToolbar        cstring(21)
szSMTPServer               cstring(256)
szSMTPAuthUser             cstring(256)
szSMTPFrom                 cstring(256)
SMTPtlsType                long
SMTPPort                   long
szBackupPath               cstring(File:MaxFilePath)
lOtherOptionFlags          long
lOtherOptionFlags2         long
                end
```

Efficient declarations

The typical method of propagating global data declarations throughout a multi-DLL project is to use the AppGen's global data store in each APP. Here's a time saving suggestion you may not have previously considered: if you don't need the data available to the AppGen (primarily the window designer and the templates), save some time and declare these data in an include file. When a project wide change of such data is required you can edit it once using your favorite editor, then simply rebuild. If you have data that has to be available to AppGen, but only in a limited number of places, use the AppGen to declare that external data and use your include file everywhere else. You will save time and minimize the possibility of errors.

Taking my own advice, I used to declare my globals in an include. Of course, any change to the group required a rebuild of the entire project. I had, after all, modified global data that is referenced in every module. Or had I?

Don't Waste Your Time

Consider this: If I change the length of a cstring in gUserOptions I have to rebuild the *entire* project, even if that cstring is directly referenced in code in only two modules. Sometimes it is simply necessary to endure such a process, but not always.

At some point during the iterative process of watching the AppGen churn through the regeneration of nearly 20 APPs I began to resent the excessive idle time required due to a simple change in a global data structure.

During one of these mind numbing rebuilds it dawned on me that I could effectively hide the underlying data of gUserOptions from the entire project, with the exception of the data DLL. The actual data would only be declared there, and need not even be exported.

Who() Can See What() I See?

The cosOptions class allows read/write access to simple data types (numeric and string types) within a group. Most importantly, cosOptions doesn't need to know anything about the GROUP's structure ahead of time, thanks to WHO and WHAT.

The Boomers among us may remember the 1963 movie "The Man with the X-Ray Eyes". Well, Clarion's WHO and WHAT offer something like that, absent all the personal issues Ray Milland endured in the movie.

WHO takes a group and an ordinal number and returns the name of the specified group field, while WHAT returns the actual contents of that group field. Using these two statements, cosOptions determines the number of fields in its group and provides read/write access to those fields. The FieldCount property is used in its internal FieldCheckSanity and FlagCheckSanity methods. What() is the workhorse that is used to access individual fields based on an index value parameter. And that brings me back to global data.

I said that the cosOptions Class requires you to pass the ordinal value of the field's position within the group. You *could* just use numbers, but that would be unwise, as it would make your code subject to errors if the numbers change, and it would also make your code somewhat illegible. So you do need some information to be available globally, in the form of EQUATEs.

It's true that if your equates change you'll need to recompile any modules that INCLUDE the source containing the equates. But all other changes, such as data types and string length, will not trigger a global recompile. Maintaining an ITEMIZEd list of equated constants is a small price to pay for the convenience you receive in return.

An easy method to create the required field index constants is to copy/paste the group fields into an ITEMIZEd equate block. *Be careful to retain the relative position of the equates to the group fields.* Retaining the group field names has the

added advantage of allowing you to search your source for their usage.

```
itemize
BRWSSCROLLCTRLTYPE      equate
SHOWRECORDTOOLBAR       equate
SMTPSERVER              equate
SMTPAUTHUSER            equate
SMTPFROM                equate
SMTPTLSTYPE             equate
SMTPPORT                equate
BACKUPPATH              equate
OTHEROPTIONFLAGS        equate
OTHEROPTIONFLAGS2       equate
  end
```

Changes are Need To Know Only

Once you've implemented the cosOptions class, making a change to the group structure is simple. For example, I have decided that my cstring(256) for gUserOptions.szSMTPAuthUser can be shortened to 129. If this change requires no modification to dependent code, then I recompile my data DLL and I'm done.

A Look Inside

The cosOptions class is not at all complex. Its functional Init method requires only a group pointer and offers an optional debugging flag. Class debugging is implemented using the Windows API's OutputDebugString() function (you can see trace results with a utility such as DebugView). The Init method is called in the data DLL, the single place where gUserOptions is in scope

```
Init procedure( *group groupIn, long Debug = 0 )
```

(There is a second Init method that accepts a *cstring; this method is intended to support read/write to INI or XML files but is at present unimplemented.)

The purpose of the five data access methods is evident from their names. ValueSet() and ValueGet() work with any simple data type (nested groups are not support by the class). The Flag methods operate on a long in the group used for bit flags. There is no Kill method implemented as there is nothing for a Kill method to do.

```
IsFlag      procedure( long FieldIndex, long Mask ), long
SetFlag     procedure( long FieldIndex, long Mask ), long
ClearFlag   procedure( long FieldIndex, long Mask ), long
ValueSet    procedure( long FieldIndex, ? ValueIn ), long
ValueGet    procedure( long FieldIndex, *? ValueOut ), long
```

Using the cosOptions Class

Lee White has contributed a global template that takes the pain out of class definition for DLL export and import. His Clarion Magazine article A Template For Exporting Classes demystifies that bit of magic.

The modified people.app included in the source download illustrates an implementation of the class using the gUserOptions group referenced in this article. Copy the cos_options.clw and cos_options.inc into your C6 LibSrc directory. The class is written in such a way that it will not be linked unless it is used. Register the included cos_options.tpl and have a look at the example. Build PeopleGlobals.APP first, then copy the LIB and DLL up one level to the People.APP directory before building that application.

Future revisions of the cosOptions class are available at the Clarion Open Source project at Google Code. Anyone that would like to contribute to the class, or to the Open Source project, is welcome to contact me.

Download the source

Blown away from South Louisiana by the winds of change, Mark Geisinger, the creator of the "How Pork Fat Can Improve Your Life" diet, has been developing software with Clarion since 1993. In addition to software development, he is a wanna-be philosopher and is patiently awaiting the call from HBO to get his stand-up comedy special kicked off (the air, likely).

Reader Comments

Posted on Thursday, October 23, 2008 by Vince Sorensen

Very nice...

BTW, is there any reason why "ValueSet" is not "SetValue", matching "SetFlag"'s "verb-noun" order?

.....

Posted on Thursday, October 23, 2008 by Mark Geisinger

Vince,

Thanks. Regarding the name ordering, it should be as you suggest. I tend to be picky about names and sloppy about naming conventions. :}

Mark

.....

Posted on Thursday, October 23, 2008 by Mark Geisinger

There's a bug in the code I need to point out.

In each of the .Init methods self.FieldCount comes out of the loop incremented one past the actual field count.

Add

```
self.FieldCount -= 1
```

as the next line after the loop in each .Init method to fix this.

Mark

Add a comment

Clarion Magazine

Filtering Reports With The Pause Control

by Steven Parker

Published 2008-10-21

The Pause Control template is probably my favorite template. When I create a new Process template procedure, I almost always populate this template on the progress window before doing anything else. Rarely do I have to go back and remove it.

> NOTE: If a report is "a Print in a loop," a Report template procedure is a Process template procedure with a report structure and a Print. So, I will use "Report" and "Process" interchangeably here. In fact, in 6.x - though not in 5.5 - the Report button is available from a Process' properties window.

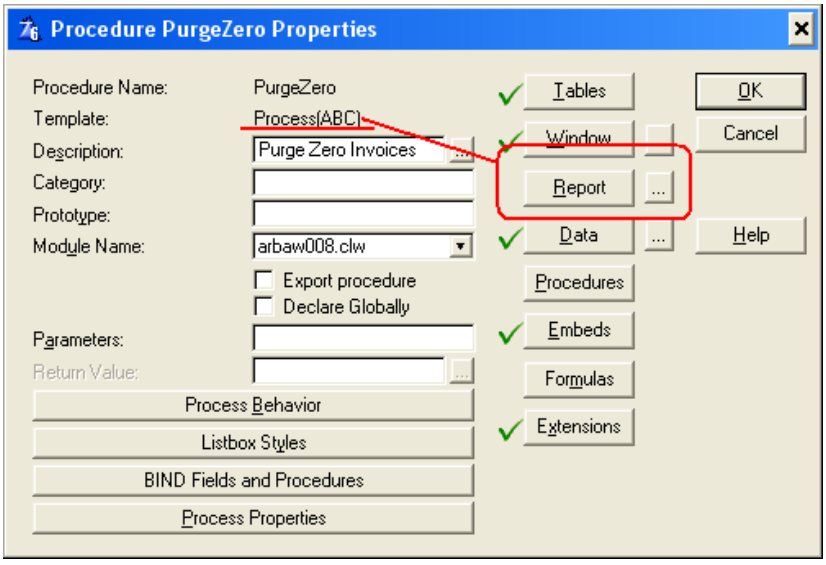


Figure 1. Report structure is available in 6.3

Typically, I collect a range of dates or customer numbers or item numbers before running a process. These data are then used in a SetFilter statement.

But this approach necessitates a separate Window procedure. With the Pause template, I do not need a second window; I can use the default Progress window and start the report in the "paused" state, waiting for input from the user.

Figure 2, below, shows the pause window with two date entry fields added.

Naturally, my users are scrupulous about entering the requested data carefully. Yeah, right, sure.

Of course, I get the call: "I entered a beginning date and an ending date; I *know* that I have customers for those dates but your program gave me a 'no records to process' message."

On closer examination of Figure 2, you will find that the user is entering the dates in the wrong order.

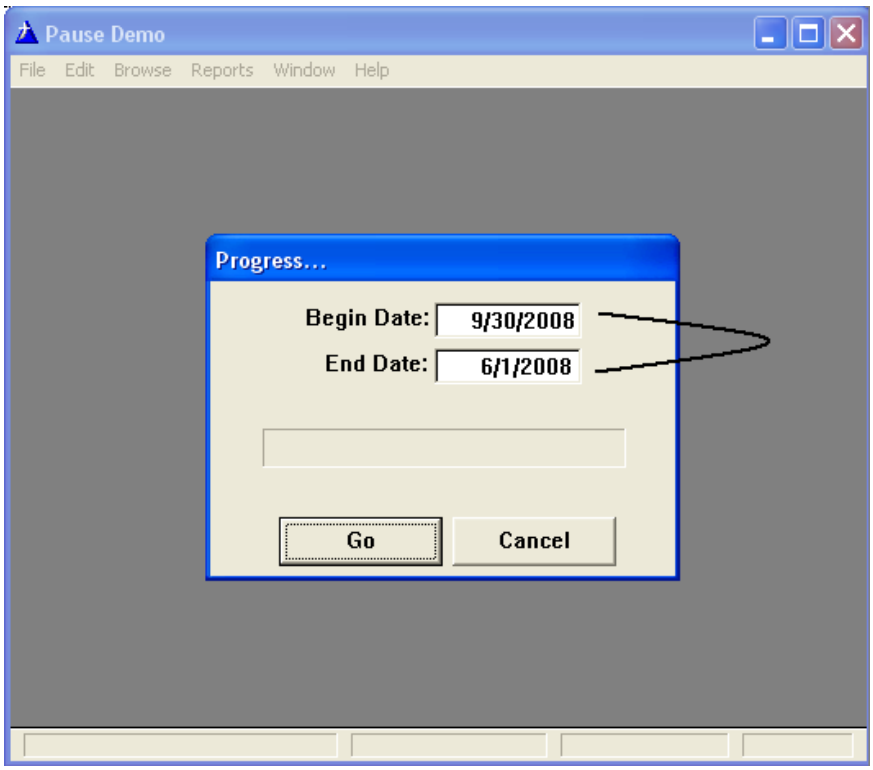


Figure 2. "Power User" date entries

Note how the "Begin Date" is later than the "End Date." To duplicate what this user does, open the demo app (locally compiled for C5.5), downloadable at the end of this article. From the main menu, select Reports | Test Reports | By Name. Enter the dates in the wrong order. (The app is an Application Wizard based on the Pause.DCT, so there is a browse where you can create test data - though I have supplied some default data with at least one record for each month in 2008. I've also included some default reports created by the wizard.)

To catch this condition and alert the user, I insert code in the Pause Button, Accepted embed:

```
If BeginDate > EndDate
  Message('Beginning date should be earlier than the ending date.'|
    ,'Error',ICON:Exclamation)
  Clear(BeginDate)
  Clear(EndDate)
  Display      ! force refresh
  Select(?BeginDate)
  Cycle
End
```

Sure enough, the next time the user enters the dates backward, the warning appears:

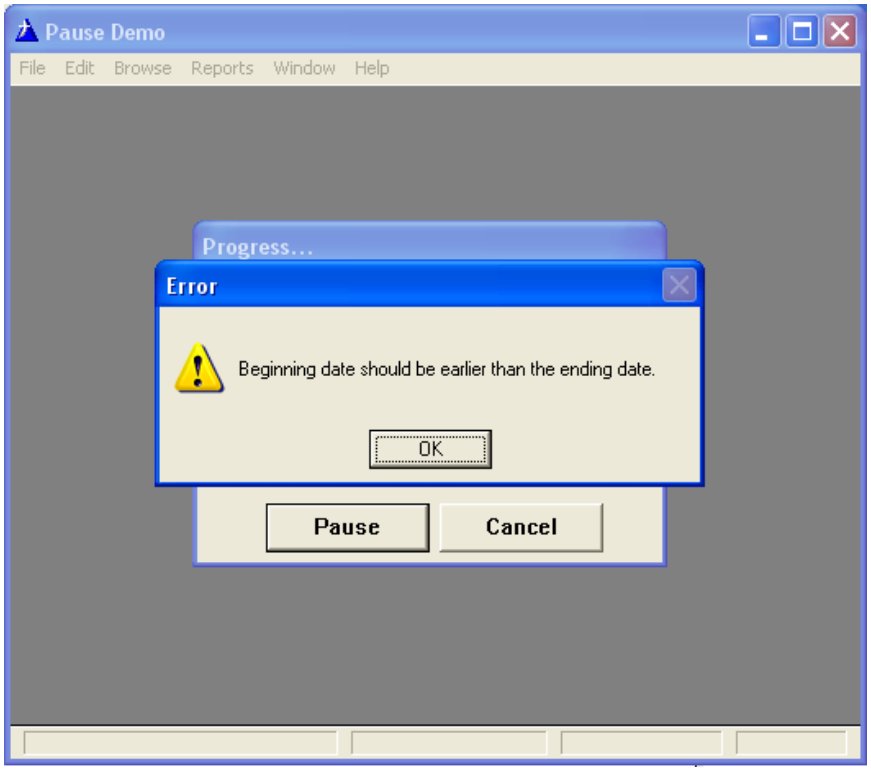


Figure 3. Warning successfully produced

However, the report still runs! And the filter, of course, still fails. The "no records" message is still presented.

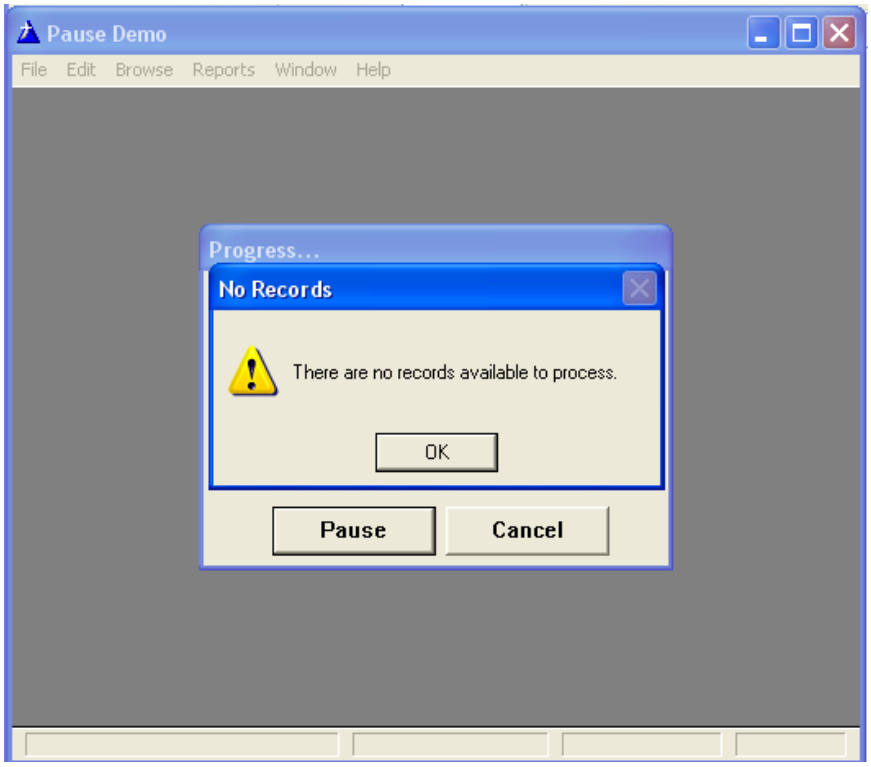


Figure 4. After the Warning

I could easily make the case that this is even worse than the original state of affairs.

What I expect from the Cycle statement, in the code above, is to be returned to the Progress window. I expect the beginning date field to be selected and both entry fields to be blank. Instead, the report continues to run.

What I expect can be seen in the demo app. From the main menu, select Reports | Test Reports | Corrected. Enter dates in the wrong order. You will get the message and you will be returned to the progress window. *This* is what should happen.

Needless to say, the default behavior is surprising.

As it turns out, the problem has to do with how reports are printed. Report procedures are primarily made up of two classes, an instance of ReportManager, called ThisWindow, and an instance of ProcessClass, called ThisReport. Oh, and then there's the report structure itself.

ThisReport takes care of opening the file, applying filters, setting sort orders and retrieving records.

ThisWindow, which as I said is an instance of ReportManager (which is itself derived from WindowManager), masterminds the whole process. Among other things it tells ThisReport which sort orders and filters to use (based on my template choices). ThisReport also opens the progress window and handles any window events, including timer events.

Timer events are events that fire at predetermined intervals; you can trigger these with a TIMER attribute on the window. And report progress windows have timer events.

When ReportManager gets a timer event, it executes a bunch of code including the following:

```
LOOP WHILE SELF.Process.RecordsProcessed - StartOfCycle < SELF.RecordsPerCycle
  RVal = SELF.TakeRecord()
  IF RVal THEN RETURN RVal .
END
```

Among other things, the ReportManager.TakeRecord method calls this bit of code:

```
RVal = SELF.Process.TakeRecord()
```

SELF.Process is a reference to the ProcessClass instance, ThisReport. And ThisReport's TakeRecord method is the one that actually prints the lines of the report.

Since printing is determined by time events, it appears that as soon as the Pause control receives what it sees as a resume command (a click on the button in question) it begins firing timer events and printing the report. In simple language, both the window and the process are inside a single Accept; that is why pressing the button "resumes" processing.

Certainly there must be a way to intercept a data entry error and stay on the window without causing the report to run.

Comparing the generated code of a process with a Pause control to the code generated by a process without it, reveals three variables populated with the Pause control template.

```
KeepVisible
DeferOpenReport
Paused
```

N.B.: The implementation of the Pause control is very different in 6.x and 5.5. In 6.x, these three variables are declared in abreport.inc and used in abreport.clw. In 5.5, the Paused variable is found in ABReport.TPW. Likewise, how the variables are used differs. I leave it to the reader to trace these variables and their use through their respective Clarion versions; it is an exercise well worth the effort.

In 6.x, when I populate a Pause control, the following code is generated into the Init method, at the indicated priorities:

```
! [Priority 8560]
! embed available here
```

```
ASSERT(~SELF.DeferWindow) ! A hidden Go button is not smart ...
SELF.KeepVisible = 1    !! not used in 5.5 !!
SELF.DeferOpenReport = 1
SELF.Timer = TARGET{PROP:Timer}
TARGET{PROP:Timer} = 0
?Pause{PROP:Text} = '&Start'
SELF.Paused = 1
?Progress:Cancel{PROP:Key} = EscKey
! [Priority 8800]
   ! embed available here
SELF.SetAlterts()
```

Note that KeepVisible is not set at this point in 5.5 code. Though the KeepVisible variable is declared in 5.5's report template, it does not appear to be used (at least, I can't find it in my demo app).

Tracing the variables' usage, I see that in both 5.5 and 6.x the Pause button, Accepted embed contains this line:

```
SELF.Paused = 1 - SELF.Paused
```

which toggles the value of SELF.Paused. This code suggests that the Paused variable determines whether or not the progress window is considered "completed" and the main processing loop may begin. If so:

```
If BeginDate > EndDate
  ?Pause{PROP:Text} = '&Start'
  Message('Beginning date should be earlier than the ending date.'|
    ,'Error',ICON:Exclamation)
  Clear(BeginDate)
  Clear(EndDate)
  Select(?BeginDate)
  Display          ! force refresh
  Self.Paused = True
End
```

should accomplish what I need. Indeed, as the demo app shows (Reports | Test Reports | Corrected, the procedure label is ByName2) shows that this does work the way I want it to. Note that I have to reset the ?Pause button text back to 'Start' as the prior generated code sets the button text to 'Pause'.

As I said, tracing the use of these variables is a worthwhile exercise.

But wait! There's more!

Three variables, KeepVisible, DeferOpenReport and Paused, control the features of the Pause control.

KeepVisible, in 6.x, seems to control display of the preview window.

DeferOpenReport, when set, prevents Self.OpenReport. This would be required if the "Go" button (I usually change the button text to "&Start" and I always tick "Start Paused") on the Pause control is not pressed.

Pause (the variable), as I showed, prevents the report from running.

These variables, then, appear to give me control over the Pause control.

Dual purpose reports

Here's a hypothetical situation: One of my End Of Day reports is a Customer Sales report. When I run End Of Day reports, the user must select a report date. I pass this date to each of the selected report and create my filter as:

```
ThisReport.SetFilter('<fileDateVariable> = pDateToPrint')
```

Elsewhere, I have the same report except that it is called from a generic Customer Reports Menu and allows the end user to get a report on a range of dates (remember the EOD version reports on one date only). In this case, I do not know the date(s) in advance, so I use the Pause Control to stop and get a range of dates from the user. My filter looks like:

```
ThisReport.SetFilter('CUS:SaleDate => LOC:BeginDate and ' & |
  'CUS:SaleDate <= LOC:EndDate')
```

It would be very nice if I could use one report for both purposes. What I have in mind is changing the report so that it takes two parameters:

```
CustomerSalesReport(Long pBeginDate,Long pEndDate)
```

My End Of Day report, for example, would be called with the same date in both parameters. If I assign the parameters to local variables in Init:

```
LOC:BeginDate = pBeginDate
LOC:EndDate = pEndDate
Or, even,
LOC:BeginDate = pBeginDate
LOC:EndDate = pEndDate
If LOC:EndDate = 0
  LOC:EndDate = LOC:BeginDate
End
ThisReport.SetFilter('CUS:SaleDate => LOC:BeginDate and CUS:SaleDate <= LOC:EndDate')
```

would work correctly for both uses.

When the report is called from End Of Day, I already pass a date. However, if the report is called with pBeginDate (or both pBeginDate and pEndDate) blank or zero, I can safely assume that it is called from the Customer Reports Menu.

When called from the Customer Reports menu, I want to stop and collect the date(s). When called from End Of Day, I want the report to start without pause. In other words: when called with the parameters blank or zero, I want the report to pause.

To reiterate: if parameters are non-zero, run without pause. Otherwise, stop and get the dates.

Suppose I create my report to start paused, *my* default behavior. Suppose I prime local variables early in Init. It follows, then, that if the parameters are non-zero (running from EOD), I want to by-pass the default, paused, behavior and simply run the report. In 6.x, my notes indicate that in Init this code does the job:

```
! [Priority 8800]
If LOC:BeginDate <> 0
  Self.Paused = 0
  Self.KeepVisible = 0
  Self.DeferOpenReport = 0
End
SELF.SetAlterts()
```

And, yes, all three variables must be zeroed:

- Self.Paused = 0 means no, I do not want to pause
- Self.KeepVisible = 0 means yes, I do want standard preview behavior
- Self.DeferOpenReport = 0 means I do want normal report opening

Clarion 5.5 gave me trouble with this simple solution, perhaps because one of the variables is in the template, not the process class. I settled on the tactic of wrapping the default generated code in a condition check (Figure 5).

```
Previewer.AllowUserZoom=True
! [Priority 8560]
If BeginDate <> 0 and EndDate <> 0      ! must be set before template
  Self.DeferOpenReport = 0
  Self.Paused = 0
Else
ASSERT(~SELF.DeferWindow) ! A hidden Go button is not smart ...
SELF.DeferOpenReport = 1
SELF.Timer = TARGET{PROP:Timer}
TARGET{PROP:Timer} = 0
?Pause{PROP:Text} = '&Start'
SELF.Paused = 1
?Progress:Cancel{PROP:Key} = EscKey
! [Priority 8800]
End
! Prepare Alert Keys
```

Figure 5. Implementation for 5.5

In the demo app, Reports | Test Reports | Pass Dates shows this. I set two global dates and call the report. Early in Init I assign the globals to locals:

```
BeginDate = GLO:Begin   ! pBeginDate
EndDate = GLO:End       ! pEndDate
```

(In the real world, I would not use globals, I would assign the parameters to local variables; I used globals here so that I could copy an existing report, the report from "Corrected," and use it to emulate the one-report-two-interfaces idea.)

Figure 5, above, shows how these locals are used in 5.5 to by-pass the Pause control in this case.

I should note that if I had not assigned the globals/parameters before the procedure call, on assignment in the report/process, the locals would contain zero. That is, because the default value of a Long is zero, not assigning a value does not result in a null parameter. This is important. Zero is not null and, therefore, I don't need to make myself crazy with Clarion's handling of omitted parameters on class methods and the checking thereof.

In sum: tracing the use of the Pause control's variables *is* a worthwhile exercise.

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.

Reader Comments

Posted on Wednesday, October 22, 2008 by Paul Howard

"Published 2001-05-01"? <g>

Nice article on using the Pause control template.

Filters are great because they use the view. Hand constructed Filters are buggers because they typically don't generate compile errors even though they can easily fail at runtime due to typos/binding issues/variable name changes, etc. :(

You'll run a lot of folks off the bridge if you don't mention that any range limits are set in .Init so when your user resets those range limit vars via paused Progress screen, they will be ignored! My solution: copy the template created AddRange to just prior to parent.OpenReport, then remove from template prompts (may not be required). The range limits will include the values set on the paused window.

Please feel free to hammer me with an easier solution.

Best Regards,
Paul

.....

Posted on Wednesday, October 22, 2008 by Paul Howard

Don't quite know how the double posting happened - sorry!
Please delete.

BTW, in c55 (and c6 IIRC), I just validate in Pause control Accepted embed before generated code, issue message, select control with invalid data, and cycle. This prevents continuing and doesn't turn on TIMER until error is fixed.

Best Regards,
Paul

.....

Posted on Wednesday, October 22, 2008 by Dave Harms

Paul,

Duplicate deleted. Not sure how you got the oddball pub date - looks fine from here.

Dave

.....

Posted on Thursday, October 23, 2008 by Paul Howard

Dave, it's right above me <g>

"Tell Us What You Think

Published 2001-05-01"

HTH,
Paul

Add a comment

Clarion Magazine

IP-Enabling An Existing Application

by Dermot Herron

Published 2008-10-17

This article was updated on Oct 17 with further information regarding ERROR(27).

This article is a simple "How-to" description of the process of making an application work with the IP driver. It expands on the description given in the manual, noticing the mistakes and misreadings that I made in the beginning. Do it my way until you are comfortable and then do it a better way!

Why bother?

Many people have had problems accessing Topspeed files over a Windows network. The problem is often the built-in shared-file-locking of Windows, which can make opening and accessing a file a *very* slow process. There is also a reliability problem because the files are open over the network and vulnerable to network failures like power-failures or unplugging a network cable.

Finally there is the network traffic problem. On lightly loaded networks the IP Driver is as fast as normal TPS access on a local machine. On a heavily loaded network, the IP Driver sends and receives much less data so application performance improves. (You can also use the IP Driver over a WAN but much depends on the quality and throughput of the WAN.)

What is the IP Driver?

The IP Driver is more than just a driver; it also consists of a developer-generated server application that runs where the files are kept. Your end-user applications no longer access your files directly; instead they send file requests to the server program which then looks up the data and returns it. Your applications can access the IP Driver server application over a local network, an intranet, or the Internet.

The IP Driver allows Clarion Topspeed files (and all other Clarion-driver-supported types of files) to be accessed in a way very similar to SQL client-server, thereby addressing speed and reliability issues. (The IP Driver even works with the SQL drivers although no one *would* do this!)

For more information on how the IP driver works see "An Overview of the IP Driver, Parts 1 and 2" in ClarionMag 31/01/2005 by Bruce Johnson. My article owes a lot to those articles.

Versions

SoftVelocity has finally got the IP Driver working very well with the latest version. The driver comprises two files supplied in a single zip file - the files I'm using are IP Driver-Server-ver63-82_bld9055.zip which contains these two files:

- IP-Server-ver63_2_3-82_bld9055.exe
- IP Driver-ver63_2_3-82_bld9055.exe

The IPServer installs all the parts of the *server* side of the equation - the programs to make the ipreq.cfg file (more on that later) and the templates and programs that go into making the server run and making the dictionary-DLL that the server uses to access the files. This is the "server application program" mentioned above.

The IP Driver installs the templates that are necessary for the remote or "slave" programs that access the server and converts all your files (automatically) to IPDRV file types.

How does the IP Driver work?

Keep *very clear* in your mind that there are TWO programs - the client and the server - I initially had trouble understanding which program the IP Driver manual was talking about at any given moment.

The two IP Driver systems permit the creation of separate client and server applications in which all the actual file opening and handling is *only done by the server program* and the communication between the two programs over the network is only done with IP commands/packets. Thus someone tripping over a cable will merely cause a request to be lost, not a vital series of file commands to be interrupted.

Contrast these file access techniques:

In non-IP Clarion applications running on a network:

> Low level commands are sent from the client machine *directly* to the target machine operating system (which acts as the "server" for the files) and thence to the files on a network-shared hard drive. These requests result in the files being accessed and manipulated by the client *over the network, at the lowest level* - files are opened/closed/updated/written-to etc. at the lowest detail level over the network. If a network cable is unplugged or a power-fail happens at the wrong moment, the file can be left open or damaged. And a lot of network bandwidth is used.

In IP Driver enabled applications:

> There is a program (IpReq.exe) which runs on the server and answers IP request packets over the LAN/WAN from the client-program. The server listens on a specific port for requests from the client (2339 for normal and 2340 for SSL programs).
>
> Thus only the server program accesses the files on the local disk when it receives IP requests from the client. The server program returns the requested data using IP to the client. Thus if a network is suddenly unplugged or the client computer switched off at the wall, no damage to the files is possible because the files are never exposed to the remote client on the network.
>
> The very clever bit is that the IP-server application template creates a DLL from your dictionary which is used by the IP-server software (IpReq.exe service). This allows the IP-server to "know" about your databases and to honour requests for them. IpReq.exe uses ipreq.cfg to "know" what DLL, tables and permissions are specified for the active client.
>
> On the client side the template automatically replaces the drivers (TPS etc.) with its own driver (IPDRV), and this driver then automatically sends appropriate requests to the server instead of to the files themselves. The driver also identifies which DLL it "owns" so the server (ipreq.exe) can look for that DLL and use it to address your files correctly.

What do you need?

There are two distinct parts to the process of IP-enabling an application. The installation programs are IP-Drive.exe which is used to install the client requirements into Clarion and IP-Serve.exe which installs the server components. (*Be very careful* that you have the *same upgrade level* for Clarion and the IP Driver - you could otherwise end up chasing very elusive fairies for a very long time!)

IP-Drive.exe installs a file driver (c60ipd.DLL) into Clarion6\bin; this driver enables the compiled client program to send requests to the server program over the IP network. The installer also registers a template (IPDRV.TPL). The template *automatically* converts all the file drivers in the client program to IPDRV. (Disabling the template with one mouse-click converts the drivers back to what they were.)

IP-Serve creates a new ClarionDataServer directory. This directory defaults to the C root drive, but I found it more useful to install the data server under Clarion6 instead for ease of access. The Server-DLL application uses the IP Data Server template which is in IPSERVER.TPL.

> **Tip:** A single computer can use the IP Driver in a client-server configuration with both the client and server running on the same computer, and you do not have to have a network. In other words, the same application can work on a single computer or over a network using the IP Driver. *You do not need to have two versions of your program.*
>
> **Very Important Tip:** Do *not* use FileManager3 in an IP-enabled application. It is possible to do so, but unnecessary and a bit of a nightmare to decide whether it is FM3 or the IP driver which is causing you trouble. Rather, build a tiny app

containing FM3 and *not* the IP Driver, with one window that opens and closes *all* your files. Run this program in the same subdirectory as the files after an upgrade, from within the installer program - you only need to run it once per upgrade. (You can use this same program to easily do other clever upgrade or maintenance tasks.)

IP-enabling the People App

Here's the overall sequence of events for IP-enabling the People APP:

- Create and generate IP_People.DLL
- Register IP_People.DLL with the server
- IP-enable the People.App by including the necessary templates.

Now the details...

Create and generate IP_People.DLL

There must be one IP_*DictionaryName*.DLL for each *dictionary* in your system. Name it after the dictionary rather than the application.

SoftVelocity recommends you name your IP Driver dictionary DLLs using the prefix "IP_". The driver registration program, will check the actual DLL files to determine which it should register. This DLL must be linked in *standalone* mode (the default) - it definitely fails otherwise.

The IP_*DictionaryName*.DLL can be compared to the data dictionary DLL in a multi-DLL program. It is created in similar manner. The steps are:

- Create a new app with the name IP_People.app
- Identify the People dictionary.
- Clear the First procedure field in Application Properties.
- Set the destination type to DLL
- Make the application template "IPServer" (important!)
- Make sure the "Application Wizard" is ticked. Click OK to run the wizard (Figure 1). All the wizard actually does is to ensure that the Generate all files in my dictionary checkbox is ticked. The Next button is not disabled correctly but does nothing after this step, so you have to press Finish.

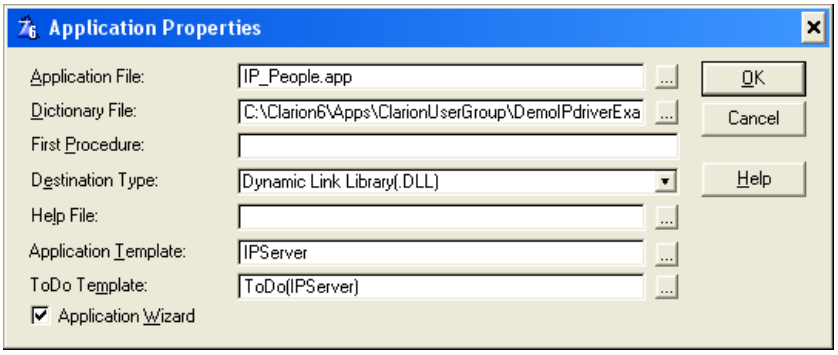


Figure 1. Running the application wizard

The resulting application has no procedures.

Generate and compile to make the IP_people.DLL. This also creates the IP_people.IPS file which is used by the client application to "understand" the DLL.

Copy IP_people.DLL to the IP-server directory (\ClarionDataServer). I do this automatically with a batch file at the end of the compile so I don't miss this essential step.

Register with the server

Next register the IP_People.DLL with the server. Change to the IP-Server directory. First run "IPSRVMGR.exe" and Remove Service, Delete Config File and then Install Service to make sure the correct service is running *out of the IP-Server directory* and *not* out of your testing directory! This is most important.

> **Tip:** The IPservice listens on the same port *no matter in which subdirectory it has been started from*. On your development computer there is likely to be more than one place where the service runs from because you will probably have more than one program and dictionary. So be very, very sure you run IPSRVMGR and execute Remove Service and Install Service whenever you change programs. This includes rmadmin.exe (see below) When (and *only* when) you are running rmadmin.exe, it is best to completely delete ipreq.cfg each time, which you can do from within IPSRVMGR. After you Remove Service use Delete Config File at the bottom before executing Install Service

Client/server interaction is controlled by the file ipreq.cfg. This config file is used to identify the files on the server to which the client application can have access and also sets the permissions for each named user. It is used by the service ipreq.exe to establish permissions for every file. A specific ipreq.cfg file is required for application distribution. It contains all the files that will be needed on the user's server (including instances of multiple applications with different dictionaries). Essentially, all the dictionaries used by an "application set" must be registered in the one ipreq.cfg.

Run the IP server administrative console rmadmin.exe inside the RMADMIN subdirectory. To connect, click on the Chain icon. Type the user name Administrator (which is case insensitive). No password is required. Leave the server as localhost:2339. (If the connection is refused you probably don't have the service running - run IPSRVMGR again and make sure the service is running).

Right Click on Data Managers in the menu tree. Then left click on Register Data Managers. All unregistered IP_*DictionaryName*.DLLs that have previously been transferred to the ClarionDataServer directory will be shown. Select the required DLL(s) with Ctrl-click and register. The selected DLL will appear in the Data Managers section of the window (Figure 2).

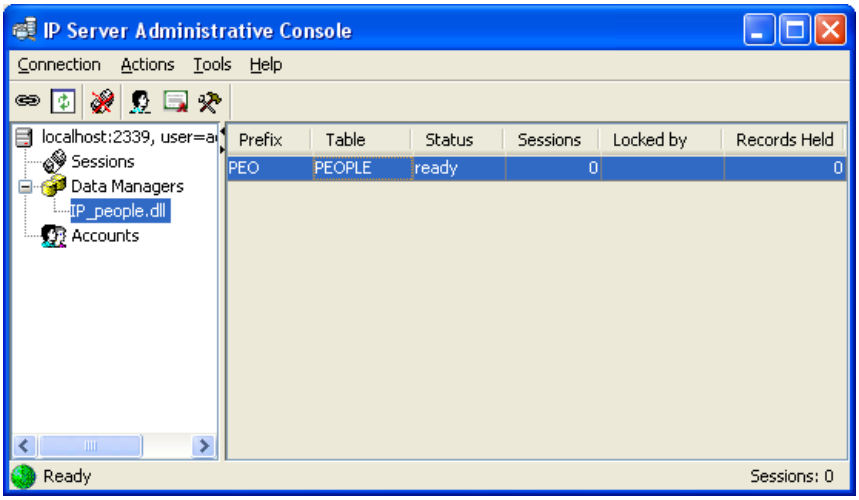


Figure 2. The IP Server Administrative Console

All registered DLLs will appear in the menu tree under the Data Managers sub branch. Click on the required DLL in this tree to view all the files in the DLL that are available to the IP Driver (in the People example there is only one file).

To check the detailed permission on a file, on the right hand side, first left click and highlight the file, then right click, and go to table permissions.

Permissions for all users are now shown (Figure 3). Note the Apply to All Tables button. Usually you will want everyone to have all permissions, but you can give restricted access to files if you need to. The default is normally correct.

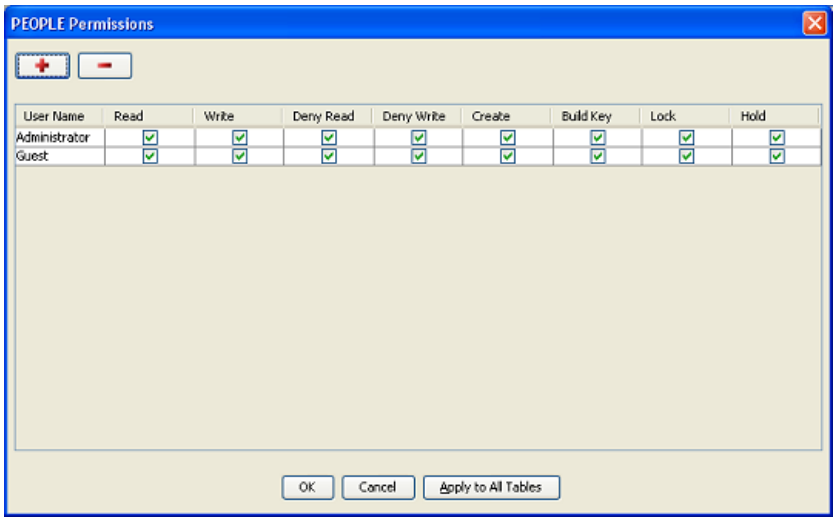


Figure 3. Permissions for the PEOPLE application (view full size image**)**

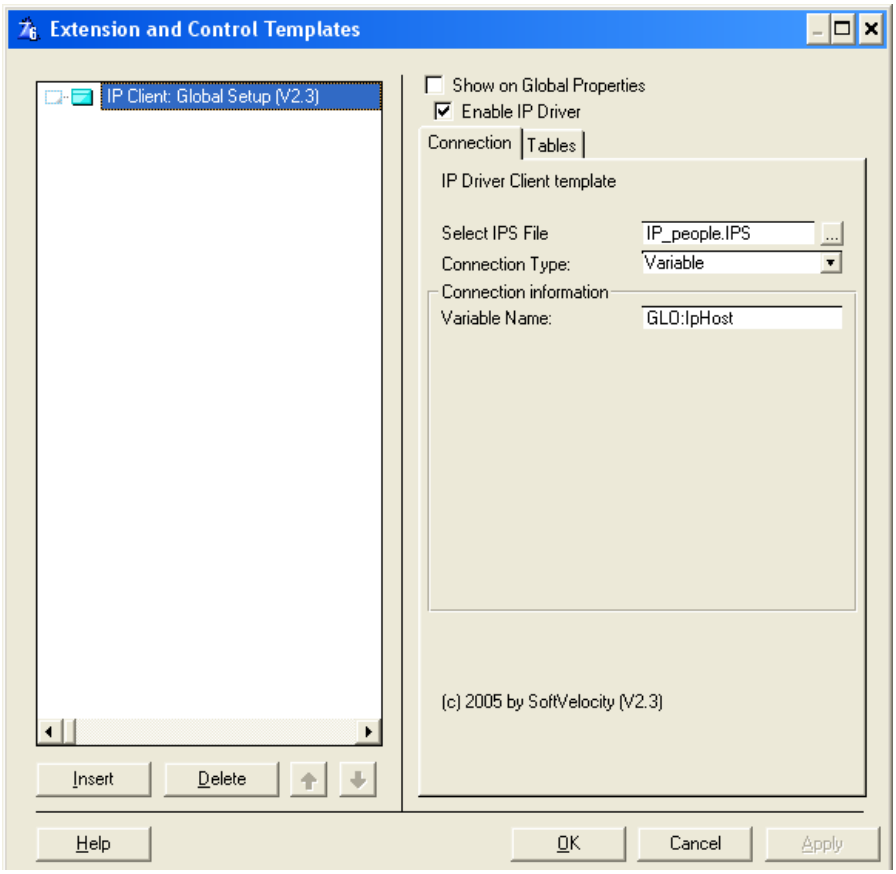
The file ipreq.cfg is now configured for all the IP_*DictionaryName*.DLLs that you registered. This preconfigured file must be distributed with the application to avoid having to teach your customers how to use rmadmin.exe!

TIP: You must remake the DLL and re-register it *every time* for the slightest change in your dictionary. This is very important and if something doesn't work, always do this before you try anything else!

IP-enable People.App

To IP-enable People app:

a. Make the app *standalone* in the Project (the default)
b. Include the template IPClient:GlobalSetup in the Global Extensions.
c. Complete the template prompts as shown in Figure 4.



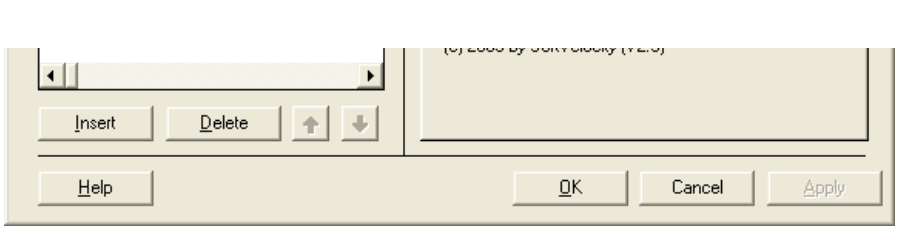


Figure 4. Global Setup template prompts

You need to specify the .IPS file which was generated when compiling the IP_peopleIP.DLL. The connection-type should be "Variable" for most applications (otherwise you have to hardcode the location of the server).

In your app you need to declare a global STRING variable about 80-100 characters long (I use Glo:IpHost) and it must be set to the correct value in the Global Embeds in Program Setup before assigning IPDRV::OWNER. On a LAN you can use the machine name (e.g. ServerA) thus:

```
Glo:IpHost = 'IP_PeopleIP.DLL@ServerA:2339'
```

Since you know the DLL name and the port (by the fact that you control both ends of the IP Driver system), the only thing needed to be supplied by the user is the machine name of the server (e.g. ServerA).

> **Tip:** At the end of the install of the server program I run a program which uses an API call to find the machine name:
>
> GetComputerName(*LPTSTR, *LPDWORD), BOOL, PASCAL, RAW, NAME('GetComputerNameA')
>
> I could display this name and ask the user to write it down but instead I write the name to an INI file on the server. When the client selects the correct subdirectory on the slave computer, the program can look up the machine name in the INI file (I use FILEDIALOG which allows LAN search). This is easier for the client than finding the computer name and then typing it in letter-perfect. My code checks to see if a valid name is kept in the .INI file, and if not it asks the user to type in the server name.

Location of everything on the server

The manual implies that there ought to be a separate subdirectory for the IP driver files. It doesn't in fact matter where ipreq.exe and its attendant files are, and it doesn't matter whether the data is in the same directory. *But the data MUST be in the same directory or in a subdirectory of the "ipreq" directory.* It cannot be on another drive. (or at least I cannot make it work on another drive!)

I prefer putting everything in just one subdirectory on the user's server-machine, including the IP files and all the DLLs the user needs and all the TPS data files used by the application. Not elegant, but effective and safe for smaller systems!
This directory contains the ipreq.exe service program and the necessary DLLs and the data.

> **Note:** If you are testing multiple IPDRV applications in different subdirectories on your development machine you'll need to "point" the IPDRV to the right subdirectory using IPSRVMGR.exe. every time you change applications.

Run IPSRVMGR and you will see the window in Figure 5. I've annotated it with the typical tasks and sequence of events.

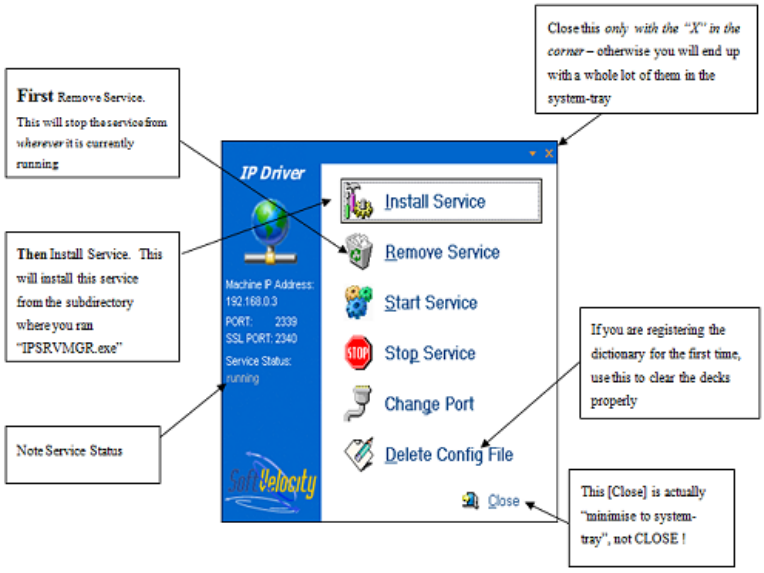


Figure 5. Typical IP Driver tasks (view full size image**)**

Things that the IP Driver can do:

- An app broken into DLLs can use the IP Driver - just follow the rules in the manual.
- Views are automatically server-side so are very fast and efficient.
- You can name the server-side files from the remote application using IPx.SetFileName(<file>, <FileName>) and so use multiple datasets. Note that at the moment you can only put data files in or below the subdirectory containing the service (ipreq.exe).
- You can add procedures to the server DLL and run those procedures from the remote application. If you have a large batch-process, this could be run where the files are, saving immense network usage.
- You can set and read variables on the server.
- You can have local files and IP Driver remote files in the same app
- You can run the IP server on Windows98SE. Although there is no proper server service in 98, it still works!

Installation of an IP Driver application at a client's site

You need two separate installation programs if you want to install a complete IP Driver based application at a client site: one for the server, the other for the client.

The Server installation contains the usual DLLs for the server to run as well as:

- IP_*DictionaryName*.DLL
- IPReq.exe
- libeay32.DLL
- OSSLWrap.DLL
- ssleay32.DLL (you need this even if not using SSL)
- ipreq.cfg (created specially for this app)
- IPSRVMGR.exe
- IPDSLicense.txt (legal requirement)

At the end of the server installation (from Lindersoft's installation program) I run

```
%INSTDIR%\IPReq.exe -i
```

which installs the service and configures it for automatic startup.

The Client installation contains your application plus the usual DLLs (e.g. C60RUNX.DLL) and:

- OSSLWrap.DLL
- ssleay32.DLL (you need this even if not using SSL)
- libeay32.DLL
- ipreq.cfg (created specially for this app)
- C60IPDX.DLL

Tip: On a single computer system you can install both the server and the client and you app will run as fast as if it were a normal non-IP app. The advantage is that you only have one set of applications which can run on any computer.

Figure 6 illustrates the various client and server components.

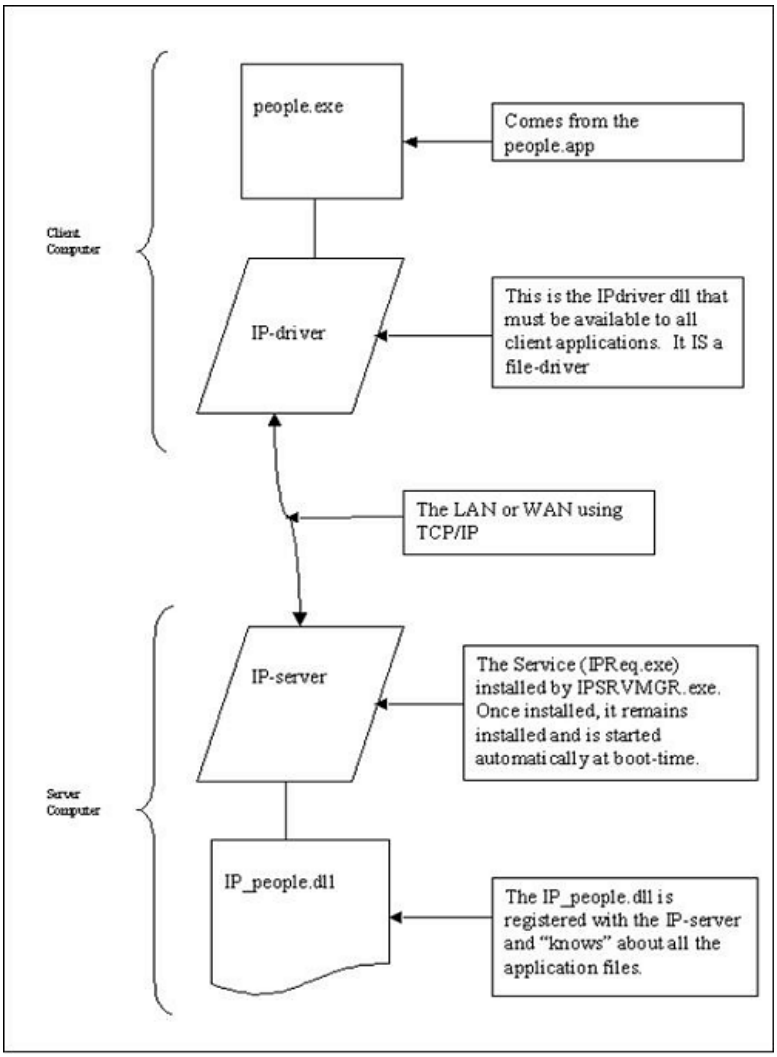


Figure 6. Block-Diagram of the IP Driver configuration

There is *very* much more to the IP Driver than is covered here. But if you can get a few simple apps working this way you will know enough background to read the very good supplied manual intelligently and go on to greater things like multi-DLL apps!

Remember to keep clear in your own mind which is the Client and which the Server.

Checklist

The following checklist assumes you have an existing APP with dictionary and wish to make it work with the IP Driver or you want to modify an already-IP Driver-based APP:

1. The Server DLL
    a. Create the IP_*DictionaryName* DLL first from the dictionary with *standalone* linking
    b. Register IP_*DictionaryName*.DLL with the server and copy ipreq.cfg to your working directory. Always do this every time you make even a slight change to the dictionary and every time you have an unknown problem!
    c. Copy the other necessary files from the ClarionDataServer to your working directory.
    d. Run "IPSRVMGR.exe" from your working directory and Remove and then Install the service
2. The IP-enabled APP
    a. Make sure the APP is linked *standalone*
    b. Include the template IPClient:GlobalSetup
    c. Make sure you set up the GLO:IpHost correctly.

Troubleshooting

If you start your application but no windows appear (that is, you can see it in Task Manager but not on the screen) and after a minute or more the application reports it cannot find the files or the server:

- Make sure the ipreq.exe is running on the server and it is running from the directory you think it is. Run IPSRVMGR.EXE from your server application directory and remove and install the service as above. (Remember that the server listens on a port (which is machine global) so is always accessible from wherever it may be running.)
- Make sure the *current version* of IP_*DictionaryName*.DLL is in the ClarionDataServer directory and cleanly re-register the DLL (i.e. delete the ipreq.cfg file). This is the bit I frequently forget even now when I make changes to the dictionary!
- Make very sure the computer name in the Client application is correct.

If you get an "Invalid file declaration" you may have:

- forgotten to recreate IP_*DictionaryName*.DLL after a dictionary change
- forgotten to re-register IP_*DictionaryName*.DLL and re-create the ipreq.cfg file after a dictionary change
- an old ipreq.cfg file (forgot to copy the ipreq.cfg file from the ClarionDataServer to your directory)
- not upgraded the file concerned with FM3 or equivalent.

Here are a few other suggestions/reminders:

Tip: Always run IPSRVMGR.EXE from within the directory where your server application must run and execute Remove Service and Install Service just to make sure that what you thought was true.

Tip: Always go through *all* the steps of registering and copying files (do it as a ritual) and stopping and starting the service - it will save you a lot of troubleshooting.

Tip: The directory containing the server needs all the files in the ClarionDataServer directory except the RMAdmin directory and the uninstall files. You must include all these files in the installation file.

Tip: Instead of using the IPSRVMGR to remove and install the service, you can use ipreq.exe with parameters:

```
ipreq.exe -u     ! will uninstall the service
ipreq.exe -i     ! will install the service
```

ERROR(27)

(The following paragraph was updated on Oct 17, 2008)

Some developers are experiencing an ERROR(27), particularly when using the IP Driver with WANs. This is a known bug

in the IPDriver and is related to threads and MDI windows. In my case I have an MDI window permanently opened on the frame. From that I was STARTing another MDI window. When I simply called that procedure instead of STARTing it my errors went away. If you're encountering ERROR(27) take a close look at how you use START with MDI windows. If all else fails consider changing as many windows as possible to non-STARTed, non-MDI. And if you encounter this bug please report it to SoftVelocity! They are working on a solution but further reports can only help move the process along.

Summary

The IP Driver is a huge step forward for developers using TPS files on a network. The reliability is enormously enhanced, the network traffic is reduced, and slow access becomes a thing of the past. There are no restrictions on the number of users. Most of the benefits of SQL (without the complications) can now be had with TPS files.

Dermot Herron grew up and went to school in Rhodesia, and earned an electrical-engineering degree at the University of Cape Town. In the 1970s he worked for the Rhodesian Post Office, which was then also the telephone company. He was given the use of an HP9100, which was the very first programmable desktop computer, to help with the design of party-line telephones. He totally fell in love with computers then and there. Leaving the Post Office, he traveled for four years, worked in Canada (as a cabinet maker) and in England (as a microprocessor engineer) and then immigrated to New Zealand. But Dermot learned that once Africa gets into your blood you are doomed! Now he runs his own company in Johannesburg, writing software to send messages to mobile phones. He lives on a 19-acre plot with a river, and spends his off-time fixing things.

Reader Comments

Posted on Sunday, October 19, 2008 by Wolfgang Orth

Dermon, thanks for your comprehensive article!

Though your article focussed on making the PEOPLE.APP IP-enabled I would like to add some comments:

- when making your first own application do not forget to set the option IPDRV = TRUE to each file in the Dictionary, otherwise you get an errorbox without any text 8-) (That bug is already reported, aehemm)
[DCT - File properties <ALT ENTER> - Tab "Options" - type IPDRV and choose BOOLEAN]

- there is a little confusion about what LIBEAY32.DLL has to be deployed. There can be found several different ones on your machine, and all seem to be wrong. It looks like SV has changed the names to IPDS_LIBEAY32.DLL and IPDS_SSLEAY32.DLL to avoid confusion with the other files, but caused even more as the *.SHP-file still names LIBEAY32.DLL and SSLEAY32.DLL to be shipped (also reported) (and even not all of the necessary Clarion-DLLs are listed in the SHP).

- Beware of the FireWall! On my dev-machine I have XP with Comodos FW. Turning off Comodos FW is not enough, I had to de-actived the XP FW also. After a while a lot of headscratching I found how to define an exception for the server to run.

Actualy just tiny hurdles, worth to overcome as the IPD is indeed a very interesting enhancement! Then its easy as pie.
Wolfgang

.....

Posted on Sunday, October 19, 2008 by Dermot Herron

Thanks Wolfgang,

I have never had to make IPDRV=TRUE for any of my apps, I only had to make it FALSE when I DIDN'T want it on a particular file. I have just checked - my current main dictionary doesn't have IPDRV=TRUE and doesn't fail. I think this needs more investigation. Maybe it is only if you create a new dictionary - I have only ever worked with existing dictionaries.

You are quite right about deploying IPDS_LIBEAY32.DLL and IPDS_SSLEAY32.DLL whatever the ship-file says. (Don't forget OSSLWrap.dll as well)

And of course you are dead right about the firewall. By the way, running 2 firewalls is generally a no-no. You MUST turn off the Windows firewall if you run Comodo otherwise interesting things happen ... <g> Comodo asks if you want to allow access and you say "yes, remember" and it then all works.

I try to convince my customers to run a firewall to the outside world but none within the lan. (It makes my job much easier ...)

I would like to re-emphasize:
1. Only use IPDriver on programs accessing data over the network.
2. Don't use FM3 on IPDriver-enabled apps unless you HAVE to (unless you are "advanced"). (FM3 certainly works - it is just confusing setting it up as well as getting the IPDriver to work. And generally you don't actually want every workstation on the LAN updating your central files.)

Dermot

Posted on Tuesday, October 21, 2008 by Stan Levine

I'd like to use this in a hand-coded application. Do you (or does anyone) have instructions on how to do this for non-app-generated applications?

Posted on Wednesday, October 22, 2008 by Michael Gorman

Question....

The Whitemarsh metabase application is strictly Client/Server. Win/32, Clarion 6.3.<latest>.

All the apps are on the Client, and the server handles all the data via ODBC with the SQL DBMS, Mimer.

Can I use the IP-Driver to enable me to make the Metabase System accessible over the Internet?

Regards,
Mike Gorman

Posted on Wednesday, October 22, 2008 by Dermot Herron

I'm ignorant of the Whitemarsh file system because I only use TPS files - my system is small enough and simple enough.

In general, if there IS a CLARION driver for the system like say TPS or DBF then the IPDriver WILL work. (As I said, it even works for SQL although one wouldn't do that.)

Dermot

Posted on Thursday, October 23, 2008 by Paul Howard

"(Be very careful that you have the same upgrade level for Clarion and the IP Driver" ... really?

More likely, 2055 works with at least c63 2055-58?

HTH,
Paul

Posted on Friday, October 24, 2008 by Dermot Herron

Thanks Paul - you are quite right - I think I should have said "Make sure you have the LATEST version of the IPDriver"

Dermot

Posted on Friday, October 24, 2008 by Dermot Herron

Stan Levine asked about a hand-coded application:

I think you HAVE to make the SERVER DLL and ipreq.cfg using the templates as per the book (or my article <g>).

On the CLIENT side, you have to:

1. include the IPDRV driver as a driver into Clarion IDE (usually done by the IPDRV install already)

2. set up IPDRV::OWNER - e,g,:

```
IPDRV::OWNER = 'IP_GolfW.dll@' & |
    CLIP(LEFT(GLO:RemoteComputerName)) & ':2339'
```

3. change all the file definitions to read, for example:

```
ErrorLog FILE, DRIVER('IPDRV'), PRE(ERL), CREATE, |
   BINDABLE, THREAD, OWNER(IPDRV::OWNER)
```

Note the DRIVER() and the OWNER()

and of course, supply the necessary DLLs to your applications.

That is all.

And the Extender and File Transfer methods you can call and use as documented.

Dermot

Posted on Monday, October 27, 2008 by Stan Levine

Thanks for the hand-coding info. I'll give it a try.

Add a comment

Clarion Magazine

More Clone(File)

by Steven Parker

Published 2008-10-07

As noted in my previous article, Clone(File), I need to clone existing file specifications in the following circumstances:

1) in a Process procedure to read a file and write it to CSV,

2) to merge two files,

3) to sequentially process a group of files.

In the last action packed episode, I showed the [Mark] Goldberg Clone. The Goldberg Clone handles dumping a file to CSV, or any other file driver, with the greatest of ease. It handles the other two needs with aplomb.

Using any pre-existing, in-scope file declaration, the Goldberg Clone creates a clone of anything containing a Record structure in just a few lines of embedded code:

```
CloneFile   FILE,DRIVER('TopSpeed'),PRE(CLONE),CREATE
Record      RECORD
              LIKE(OriginalLabel.Record)
            END
           END
```

However, if the original file has keys and I want to preserve those keys in the clone, I have to copy, paste and update (for the prefix change) those declarations (and memo declarations too). This is the one and, so far as I can tell, only drawback to the Goldberg Clone. Otherwise, the Goldberg Clone gives me the option to do a straight clone, to change file drivers, to create a new file with different keys, to do everything but change the fields themselves. (I tried adding a new field to the CloneFile structure and got bizarre results, though I didn't try very hard to debug it; I suppose this might be considered a drawback, but a minor one.)

But because I do periodically change key declarations, it would be really nice if there were a method of cloning that eliminates the need to copy and paste those.

So, soon after Mark Goldberg offered his solution to cloning, in a news group posting, S. Jayashankar, known as "Shankar," offered a template. (If you frequent the Clarion 6 news group, you will recognize Shankar as another very knowledgeable and very helpful member of the Clarion community.)

His template is available at my download center (look for "Emulate LIKE(pre:Record)" dated 8 April '06) as well as at the end of this article. The downloadable file contains Shankar's original template which cloned a file and nothing else plus variants by both him and me which take an optional NAME.

The second variant of Shankar's template requires the Label of a file-to-clone and offers an optional NAME (variable or constant). Note that this option is virtually necessary if the file(s) is not in the current directory.

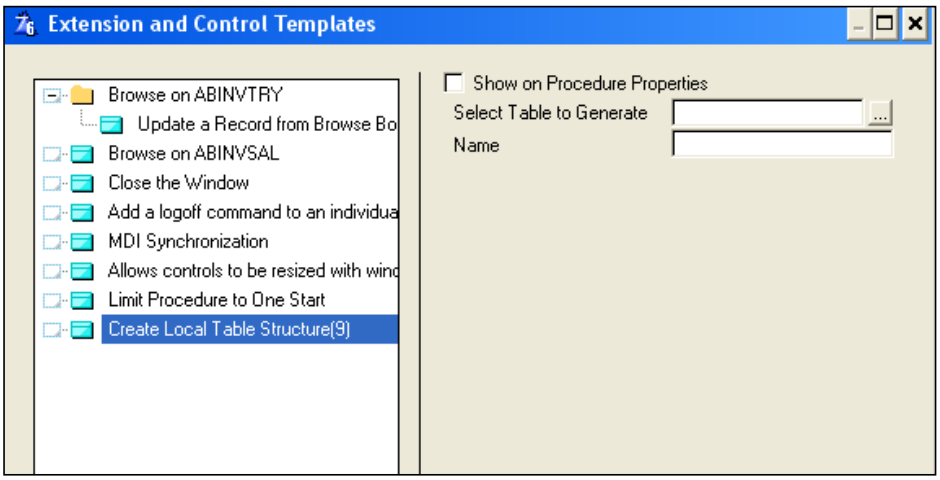


Figure 1. Clone Template Prompts

The template is provided as text. It needs to be added to an existing template chain (save it in a .TPW file and add an #INCLUDE statement to the chain's .TPL file or copy and paste into an existing .TPW or .TPL file) or turned into a stand alone template (add a #TEMPLATE(*<templateName>*,*<templateDescription*),FAMILY('ABC') header and changing the extension to .TPL).

The template is an Extension template, so after you've registered it as a TPL or added it to an existing chain as a .TPW, you need to added it to a procedure from the Extensions button.

I have successfully tested Shankar's template in both 5.5 and 6.x. And, yes, more than one file can be cloned per procedure but there is no check on double cloning a single file; this is the developer's responsibility.

The file-to-clone must be in the procedure's file schematic:

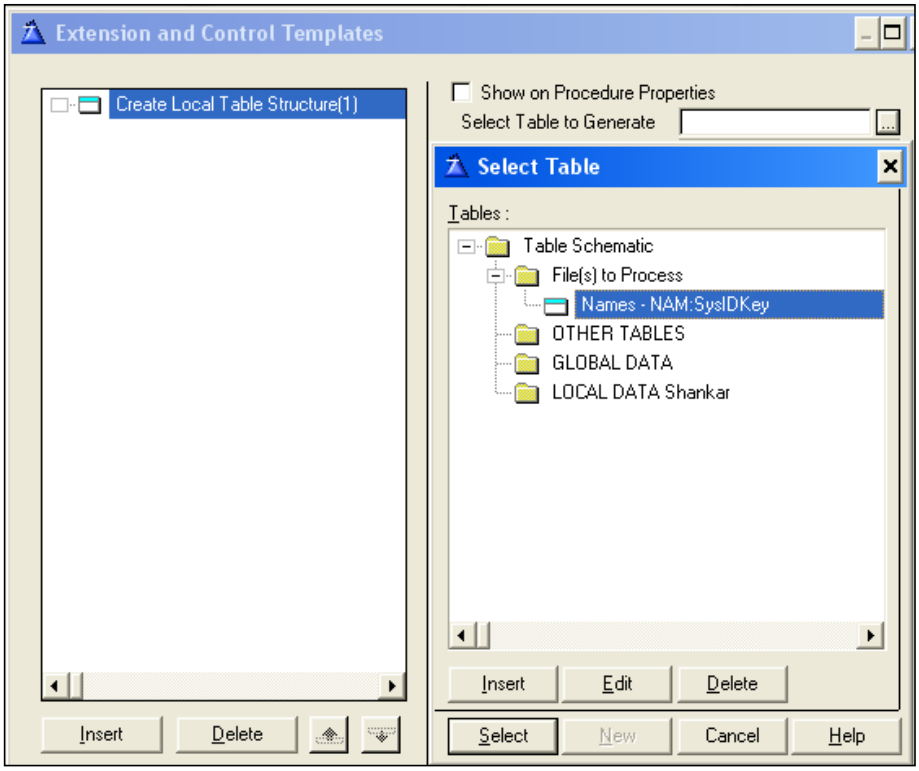


Figure 2. File to Clone Must Be in Schematic

Using the template language, Shankar's template retrieves the nominated file's specification and rewrites it to a new, local declaration. The template adds "loc" to the existing Label. It prepends an "l" (lower case letter ell) to the existing Prefix. Prepending "loc" and "l" ensure that all the labels are unique.

If the Name prompt is completed, the NAME attribute is added to the file declaration with the appropriate variable or string constant.

If I select the Names file from last time's demo app and enter LOC:TargetFile in the Name prompt, the template will generate this declaration:

```
LOC:TargetFile      STRING(100)  ! manually declared
```

(In my variant of the template I have to declare this variable myself; I believe Shankar's version does offer a lookup.)

Shankar's template will generate the following:

```
locNames  FILE,DRIVER('TOPSPEED'),PRE(lNAM),BINDABLE,CREATE, |
          Name(LOC:TargetFile)
SysIDKey    KEY(lNAM:SysID),NOCASE,OPT,PRIMARY
NameKey     KEY(lNAM:LastName,lNAM:FirstName),DUP,NOCASE,OPT
Record      RECORD,PRE()
SysID        LONG
LastName      STRING(20)
FirstName     STRING(20)
State        STRING(2)
          END
        END
```

(I have highlighted, by bold face, the text added by the template.)

I have included the generated code for a sample procedure in the downloadable source so you can see the generated code in context. If you open that CLW and compare it to the Names file declaration generated by the demo app, you will find that locNAMES, except as noted, *is* an exact clone of the original file. Shankar has delivered precisely what was needed with his template.

Merging Files with the Template

Suppose I have two Names files that I want to merge into a single file. Let's say that one resides in the current directory. The other resides in R:\MyStuff. I want to merge R:\MyStuff\Names.TPS into the file in the current directory.

I create a Process procedure to do the work. The Names file will be the file I nominate in the procedure's file schematic. I want R:\MyStuff\Names.TPS to be the source file for the Process, so I will ensure that I set Names' NAME variable to 'R:\MyStuff\Names.TPS' (I'm thinking that I could have picked a better file name for this example ...).

The "local" copy of the file, the declaration generated by the template, is the target. *This* is the "clone."

Why configure it this way? Set up this way means that I loop through and read R:\MyStuff\Names.TPS. Therefore, the Process template's progress bar will update based on how many records I've read and processed. The target copy is "just" for writing.

From the Extensions button, I select the Create Local Table Structure extension. Names is the Table to Generate. Because the target file (the clone, Names.TPS) is in the local directory, I can leave the template's Name prompt blank. Or, I can play it safe and enter 'Names.TPS' or nominate a variable and prime it to 'Names.TPS'.

> **NOTE:** Normally, I have such a variable for each file as I almost always use the NAME attribute and it is primed when the application opens; in that case, I will want to save the original value in that variable, overwrite it with the source file's path/name and restore the default value when I exit the procedure).

Finally, I need to ensure that the source file Name is pointed at the correct file, R:\MyStuff\Names.TPS, before OpenFiles:

```
LOC:TargetFile = ! copy in current directory
GLO:NamesFile =  R:\MyStuff\Names.TPS'
```

And, because the template does not manage the clone, I also need to manually open the clone:

```
Open(locNames)
If ErrorCode()
  ! create/open if appropriate or
  ! warn user if that is appropriate
End
```

I also need to be sure to close the file when I'm done.

The merge, in TakeRecord, is vanilla file processing code:

```
lNAM:SysID = NAM:SysID
Get(locNames,lNAM:SysIDKey)
If ErrorCode()              ! record does not already exist
  lNAM:Record = NAM:Record
  Add(locNames)
Else
  ! update existing record?
End
```

There is no FileManager for the clone file so I use Clarion language statements instead of FileManager methods. For the purposes of demonstration, I am actually assuming that there are no duplicate SysIDs in the two files (for the sake of argument, assume the two "locations" had different seeds for the initial SysID). Typically, I would have a location field that I would prime after lNAM:Record = NAM:Record, so duplicates would not be an issue for me:

```
lNAM:Record = NAM:Record
lNAM:LocationID = LOC:Location
Add(locNames)
```

If you wish you can add error checking after the Add but, rare as this is, it is not required here.

Sequential File Processing

In the previous article, I mentioned a customer using my Sale On Hold/Quote feature for service tickets.

This is the situation I described:

> "Sale On Hold/Quote" allows a transaction to be completed without tendering. It is "completed" by being stored in a TPS file [a completed transaction would be logged, not stored]. The file can be read back in to an active transaction and that transaction can be completed in standard ways.
>
> [One] customer puts his service calls into Sale On Hold. When the technician returns to the store, he retrieves each Sale On Hold, one after another, and tenders them out. Only, after seven or eight transactions, he gets a "File Not Open" error. Unfortunately, the File Not Open appears at a point where there is simply no explanation for it.
>
> If there were an obvious problem with how my code was written, the code would error out on the first transaction. So, I know that the problem is unlikely to be entirely of my doing. In fact, it has all the symptoms of the dreaded "threading

bug." The only problem with this diagnosis is the fact that the app was running in Clarion 6.2.

Copying the file specification and using a different NAME variable cured the issue entirely.

Now I can remove that copy of the file specification, use Shankar's template and add loc and l to the existing code that copies the data out of the TPS file and into the queue supporting the transaction (opening the Sale On Hold file, looping through it and assigning fields to queue elements is all hand code).

This, too, cures the issue entirely. However, use of the template means that if I ever change the specification of the dictionary structure defining Sale On Hold files, I won't get an ErrorCode 47 in testing or, worse, in the field because I forget to test Sale On Hold after making such a change....

Room for Improvement?

As I prepared this article a few questions arose.

Currently the template has the PROCEDURE key word. This means that a clone cannot be declared globally or at the module level. Removing that key word from the template would address this restriction, but the #AT(%DataSection) statement would need to be changed since %DataSection is a procedure embed point. ("Discuss amongst yourselves.")

The template does not open/create/close the clone. It's not rocket science to add that code. Until you start thinking about what you want to do if the clone does not exist. You don't always want to create it. This makes hand coding the open/close much less of a burden than it might, at first sight, appear.

What about an owner string? For cloning SQL tables this might be a convenience. Again, not rocket science to add.

Summary

Cloning file declarations sounds like something that should be easy. It isn't.

It isn't easy because there are diverse legitimate uses for clones and those uses are not necessarily mutually compatible. That is why I have described both the Goldberg Clone and the Shanker Shuffle (note how I've avoided "Dolly"). Between the two, my cloning needs are more than adequately addressed.

So, the final and most import word on the subject is: Msrrs. Goldberg and Shankar, on behalf of the Clarion community, I thank you.

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.

Reader Comments

Add a comment

Clarion Magazine

The ClarionMag Blog

Get automatic notification of new items! RSS feeds are available for:

XML All blog entries
XML All new items, including blogs

Blog Categories

- »All Blog Entries
- »Clarion 7 Clarion.NET
- »Future Articles
- »News flashes
- »Nifty Stuff

C7 AppGen Released To Third Party Developers

Direct link

Posted Tuesday, October 21, 2008 by Dave Harms

It's been a long, long wait. And for a small group of Clarion developers, at least, that wait is now over. SoftVelocity today released C7 with the Application Generator to third party vendors for compatibility testing.

Why hasn't AppGen gone out to all beta program participants? Releasing the AppGen to third party vendors ahead of time has two benefits. First, this testing may reveal problems that would prevent other Clarion developers who use third party products from compiling their apps; besides the possibility of bugs, the C7 compiler is stricter than the C6 compiler and some not-quite-legal-but-works-fine-in-C6 code may need to be tweaked. Second, getting the third party issues dealt with up front means less trouble later distinguishing C7 issues from third party issues. On the whole, letting the third party folks kick C7 around should shorten the overall beta period and make for a more productive beta period for C7 developers.

I should have a First Look up tomorrow (Wednesday) so I'll save most of my comments for that article. And really I haven't had a chance to do much except open a few apps and find my way around the new AppGen. I have managed to crash the IDE a few times, as have other developers, which isn't a big surprise. This is after all the first time anyone outside SV has played with this product, and we're bound to stress it in ways the SV developers never will. I do expect to see fairly quick progress on reported bugs, based on my experience with the initial IDE beta.

I've certainly been able to open an app, have it convert, use the window designer, add embed code (via embed points and the embeditor), compile and run, so I know the AppGen works, and does most (nearly all) of the things I need it to do. And it's fast. In one test I repeatedly added a new global variable to the School app, regenerated and recompiled. C6 took 17

seconds; C7 took just 10 seconds.

The AppGen overall looks familiar, if modernized. Some things have shifted place; some other things are slightly easier to get to. There's a whole lot that's new, including solutions containing multiple APPs (and PRJs), the color-coded embeditor, code folding, support for multiple versions of Clarion (which needs some work but looks promising; more on that in the First Look), a real source code editor, the file schema pad, the very cool alignment features in the window designer... well, I'll leave it at that for now. More cool stuff tomorrow.