Clarion Magazine

Clarion News

- » Clarion Third Party Profile Exchange Updated June 29 2006
- » CHT Build 10C1.0
- » PDF-XChange/Tools 3.6102
- » Software Testing Service Price Increase July 1
- » xFastButton 1.2
- » xPrettyAbout 1.2
- » xTimers 1.2
- » PrintWindow 1.04
- » WindowID 1.01 Released
- » KeepingTabs Goes Gold
- » SendTo Goes Gold
- » zAssist Updated
- » Glasso Icon Set Discount
- » Go To Lunch Compiler Updated
- » SV Newsgroups Back Online!
- » Whitemarsh Monthly Papers
- » SV News Server Expected Back on Tuesday
- » SV Discussion/News Server Down Today
- » SoftVelocity Course IV-Master's Language Series
- » dpQuery 2.08
- » Huenuleufu Releases WindowID 1.00
- » PrintWindow 1.03
- » xDigitalClock 2.0
- » View Wizard 6.02
- » Cards32.dll Demo

- » ClarionTools Spreadsheet Wizard 6.05
- » Mallorca Icon Collection
- » Gary James On The Road
- » ClarionTools CrossTab Wizard 6.02
- » Huenuleufu Lowers Initial Prices, Adds Maintenance Plan
- » Encourager Software Blog C7 Category
- » EasyExcel 4.01
- » KeepingTabs Goes Gold
- » SendTo Goes Gold
- » CapeSoft World Tour
- » EasyCOMCreator 1.04
- » Report Wizard 6.06
- » xTipOfDay 2.7
- » Class Anatomy 1.2.0
- » iAlchemy Site and Email Down
- » Clarion C6.3 Build 9053
- » J-Spell 1.14
- » FullRecord 1.53
- » IQ-XML 1.18
- » xWordCOM 1.6
- » xClarionSwitcher 1.08
- » zAssist Adds Features
- » Castle Computer Technologies Offers Testing Service
- » xQuickFilter 2.19
- » Clarion Project Manager
- » EasyVersion 3 Released as Freeware.
- » xXPframe 1.4
- » PrintWindow Gold 1.01
- » SetupBuilder 5.4 Softpedia Review

[More news]

Latest Free Content

- » SV Releases C7 Video
- » Tips Book Table of Contents Available

[More free articles]

Clarion Sites

Podcast

[Track lists, more podcasts]

Clarion Blogs

Latest Subscriber Content

Global Variables, Threads, Critical interSections and the Dangers of Unprotected Sets, Part 2

Inspired by the "CapeSoft Clowns" at the recent Aussie DevCon, John Morter sets out to create an easier way to manage thread-safe variable assignments under Clarion 6. In this concluding part John explains the class source.

Posted Thursday, June 29, 2006

PrintWindow Provides Alternative To Standard Reporting

Huenuleufu Development's PrintWindow reporting add-on provides a highly customizable way to reproduce screen layouts in report form.

Posted Wednesday, June 28, 2006

Global Variables, Threads, Critical interSections and the Dangers of Unprotected Sets

Inspired by the "CapeSoft Clowns" at the recent Aussie DevCon, John Morter sets out to

create an easier way to manage thread-safe variable assignments under Clarion 6. Part 1 of 2.

Posted Thursday, June 22, 2006

External Business Rules with the In-Memory Driver

Towards the end of 2004 Nardus Swanevelder wrote a series of articles on Clarion's Business rules, and how they could be configured at runtime. In this update, Nardus shows how to use configurable business rules with the In-Memory Driver. SOURCE LINK UPDATED!

Posted Wednesday, June 21, 2006

Using RMChart with Clarion

After researching several charting alternatives, Al Randall happened upon RMChart, a free, lightweight wrapper around Microsoft's GDIPLus graphics library. Al shows how to create charts in Clarion using the DLL version.

Posted Thursday, June 15, 2006

PDF for May 2006 (Updated)

All Clarion Magazine articles for May 2006 in PDF format. If you downloaded a previous version, please get this one. The original was missing the text of two articles.

Posted Thursday, June 15, 2006

Clarion 7 and Clarion.NET: Video No 1

Dave Harms offers a summary of and perspective on SoftVelocity's new screencast covering many of the new features in C7.

Posted Monday, June 12, 2006

SV Releases C7 Video (free article)

SoftVelocity has released a video showing the C7 XP Theme support in some detail. You can also download the PowerPoint presentation. Look for a review later today in Clarion Magazine.

Posted Monday, June 12, 2006

Tips Book Table of Contents Available (free article)

The table of contents for Clarion Tips & Techniques Vol 3 is now available for download. Editing is almost complete, with indexing to follow. You can still save $15 if you pre-purchase before the book ships in late June/early July.

Posted Wednesday, June 07, 2006

Aussie DevCon: Charles Edmonds' Pro-Series

You would think that the "Fun With Capesoft" session would be a hard act to follow, but David Beggs did it with aplomb. David presented three products by Charles Edmonds: ProScan, ProImage, and the intriguing new Clarion Code Editor. Geoff Robinson reports.

Posted Tuesday, June 06, 2006

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

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 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

Dr.Explain 2.0

Dr.Explain v.2.0 is a software documentation tool which captures windows, dialogs, and forms from live applications. Dr. Explain takes screenshots, automatically adds references to all controls, and can produce CHM, RTF and HTML help files with interactive screenshots, menus, cross-references, and navigation.
Posted Wednesday, August 02, 2006

Shareware Industry Conference 2006

Sue Pichotta reports on SIC 2006, also attended by Clarion developers Benjamin Krajmalnik, John Verbeeten and Ramon Reed.
Posted Wednesday, August 02, 2006

Using Incentive Sales

Sue Pichotta has a new article on how to make money from people who don't buy your software.
Posted Wednesday, August 02, 2006

UK Usergroup and CapeSoft Training Photos

Jono Woodhouse has uploaded the first batch of photos. One minor correction is that in the photo marked as Andy drinking his beer off the table top, it's actually frozen margarita.
Posted Wednesday, August 02, 2006

PDF-XChange PDF Viewer Beta 3

Beta 3 of the PDF-XChange PDF Viewer has now been sent out to program participants.
Posted Wednesday, August 02, 2006

CHT What's New for July

Clarion Handy Tools posts monthly "What's New" pages.
Posted Wednesday, August 02, 2006

Mallorca Collection 1.1

Another 73 new icons have been added to the Mallorca Collection, for a total of 693 Vista style icons. Sizes: 16x16, 24x14, 32x32, 48x48, 64x64, 72x72. 96x96, 128x128, 256x256. File types: ICO, PSD, PNG, GIF, BMP, JPG and ICNS. For a limited time get a $100 discount.
Posted Wednesday, August 02, 2006

SoftVelocity Summer Update

Encourager Software reminds Clarion developers of SoftVelocity's special pricing for the Core Subscription Program. The specific details for your pricing must be obtained from SoftVelocity or an authorized distributor. Special update pricing is for a limited time. This might be a reminder for some to send SoftVelocity an updated email address for contact.
Posted Wednesday, August 02, 2006

Firebird 2.0 RC 3

Kelvin Chua reports that Firebird 2.0 Release Candidate 3 is now available.
Posted Wednesday, August 02, 2006

DevDawn Report on UKCUG C7/Clarion.NET Presentation

DevDawn has a summary up on the presentation by Bob Z et al on Clarion.NET and C7 at the UK Clarion User Group meeting.
Posted Friday, July 28, 2006

SV Releases C7/Clarion.NET IDE Video

SoftVelocity has released a video presentation that Bob Z created for the Aussie DevCon. This video was late for the conference but did make it onto the post-conference CD.This first video is an overview only, but the slides cover the main features of the new IDE components.
Posted Thursday, July 27, 2006

ProcedureNotes One Day Sale

Castle Computer's ProcedureNotes template is now on sale until 11.59 pm eastern time on July 25, 2006. The templates are normally $21, but today you can purchase the template for $12. Template Highlights: Keep detailed notes about your application; Keep detailed notes about each procedure; Keep a To Do List both globally and for each procedure; Keep a revision history both globally and for each procedure; 2 Utility templates to choose from; Print the notes and information about your app; Print just the notes, To Do and revision history.
Posted Tuesday, July 25, 2006

Capesoft World Tour In Europe & UK

The final part of the world tour is about to take place in Europe, with the following destinations: Cambridge , England on 29 and 30 of July; Utrecht Netherlands (near Amsterdam) on 1st and 2nd of August; Drammen, Norway (near Oslo) on 3rd & 4th of August. The registration for Cambridge is now closed, but registration for the Utrecht leg is still open until Tuesday 25 July and for the Drammen leg until Friday 28 July.
Posted Tuesday, July 25, 2006

xFiles gets Va Va Voooom!

CapeSoft xFiles lets you write to XML and read those XML files from your application simply and really, really rapidly. The latest (and greatest) release builds on the existing support for groups and queue, and adds support for tables. Create an XML file from a table in your application with a single method call and import an existing file in the same way. xFiles now handles encoding of BLOB data, CDATA tags to allow extended character sets and line breaks to be used in fields, and more. For the duration of the European leg of the World Tour, xFiles will be at its current low price of $39, thereafter (on the 7 August) it will be increased to $59 for the beta version ($99 when the gold version is released).
Posted Tuesday, July 25, 2006

Replicate Goes Gold

CapeSoft's Replicate provides an automatic, driver independent, file-version independent, mechanism for replicating the data in two or more databases. After more than 120 public beta releases, Replicate is now at the stage where it is solid and able to be used in mission critical applications. For the duration of the UK & European leg of the world tour, CapeSoft will be keeping Replicate at the beta price of $399. On the 7 August , the gold price ($599) will be implemented.
Posted Tuesday, July 25, 2006

cpTracker Enterprise (SQL) Source Sale Ends July 31

cpTracker Enterprise Source is currently on sale through the end of July. Works with MS SQL Server, and also the new 2005 Express Edition which is free to use. There are two editions of cpTracker Enterprise (SQL): Vanilla, with most 3rd party addons removed (you must own FM3 and EasyResize); Wizard, which also includes the Query, Report and Spreadsheet Wizards. This offer also includes a site license (unlimited users for a single company) for either cpTracker Enterprise (SQL) or cpTracker Professional (TPS) full commercial editions. ClarionTools.com is offering a 45% discount if all three of the above wizards are purchased as a bundle (new sales only - no upgrade discount). 25% discount for Report and Spreadsheet Wizards if purchased individually. You must purchase these products directly from ClarionTools. CapeSoft is offering a 10% discount off File Manager 3 if purchased within 7 days of making the cpTracker purchase. Details to be emailed when your cpTracker order is processed.
Posted Tuesday, July 25, 2006

New Product: SimFileLauncher

SimFileLauncher enables you to consolidate all your help files. A single entry in the Accessories/Help menu of the IDE opens SimFileLauncher from which you can access as many help or other files as you wish. Previoiusly, to add items to your IDE menu was difficult, now it just takes a single click. With the current limitations in the IDE some 3rd Party developers have stopped adding their help files to the IDE Accessories menu. With SimFileLauncher you can easily have access to their help files right from the IDE. This applies to other files as well e.g. you can launch Carl Barnes' Source Search from SimFileLauncher. Furthermore, 3rd Party developers can take advantage of the presence of SimFileLauncher by simply adding a small text file containing their help file information to the folder where SimFileLauncher resides. Details will be available on the web site shortly. A 30 day trial version is available.
Posted Tuesday, July 25, 2006

GTL6.22

The Go To Lunch batch compiler version 6.22 fixes a bug in the "retain project subsets" feature. Available from the Par2 download center.
Posted Tuesday, July 25, 2006

Evolution Browse Export 1.11

Evolution Browse Export 1.11 adds the ability to sort the information to export, with multiple levels. All customers will receive a private email with the free download information.
Posted Tuesday, July 25, 2006

Simsoft Templates Improved

Two new templates have been added to the Simsoft Templates. The first template is a global extension template which allows one to set the color of fields with the req attribute. This lets the user know at first glance which fields he must fill in. (In a FORM the REQ is handled automatically, but in an ordinary window you can also set an input field to REQ and the color will show there too. Of course you will then need to put a cycle on your OK button when the REQ field is not completed.) The second template is a control template that lets the programmer drop a string and rounded box to a window informing the user that those fields with the color shown in the box are REQUIRED. The box takes the color from the global template. This upgrade is FREE to registered users.
Posted Tuesday, July 25, 2006

ANN: J-Fax 1.20 Gold

J-Fax 1.20 Gold is now available. Features now include:; Fax using a single line of code.; Fax Clarion 5.5 reports.; Fax Clarion 6 reports.; Fax Clarion 6 advanced reports (Adds a "Fax" button to the previewer).; Fax reports from Icetips Previewer (Adds a "Fax" button to the previewer).; Supports Fomin Report Builder!; Supports CPCS!; All of the above can be done using the local fax modem, or by changing one setting you can use a Fax Server running on another computer on the network.
Posted Tuesday, July 25, 2006

Scott & Ilka Ferrett Join Bob Z at UK Clarion User Group

Scott & Ilka Ferrett will be joining Bob Zaunere at the next UK User Group meeting to

speak on the new features of Clarion 7. Some 3rd party suppliers are also offering exclusive on the day discounts for attendees so not only will it be a great chance to improve your skills, you can also save money at the same time! There are door prizes as well.
Posted Tuesday, July 25, 2006

J-Spell 1.26 Gold

J-Spell v1.26 Gold has been released. J-Spell is now fully compatible with Fomin Report Builder.
Posted Tuesday, July 25, 2006

Fomin Report Builder 3.05

Fomin Report Builder version 3.05 is now available. Changes include: J-Spell compatibility; CPCS 6 preview with output generator buttons compatibility; Multi-Proj "Versions" feature compatibility; Query Wizard compatibility revised; MAV, FM3 and other direct ODBC tools "local" app mode compatibility; Template prompt to allow "No Records Found" message ('Always|Never|Conditionally'); Numerous other accumulated changes and fixes up to date.
Posted Tuesday, July 25, 2006

Gitano Sale

For a limited time get any of the following utilities for just $49 each: gCal; gCalc; gNotes; gFileFind; gSec; gQ; Look Good Package.
Posted Tuesday, July 25, 2006

SimPageofPage Template Enhanced

The SimPageOfPage template has been enhanced with the SimAlternateFooter and SimDifferentFooter control template, which allow you to control footer settings for left and right pages. Available for $19 from www.clarionshop.com.
Posted Tuesday, July 25, 2006

xButton Class 2.0

New in xButon Class 2.0: Rewritten library without any black box DLL; Two new code templates; Compatible with Clarion 6.x (6.3, 6.2, 6.1), Clarion 5.5 and Clarion 5; Updated

docs and install kit.
Posted Tuesday, July 25, 2006

Data Mapper 1.26

Data Mapper 1.26 is now available. The free version simply displays a nag screen as it starts up, and you're limited to saving two dictionaries and five views per dictionary at one time.
Posted Tuesday, July 25, 2006

EasyNETComponents 1.00

EasyNETComponents allows you to use Microsoft .Net FrameWork components in your Clarion 6.1 or higher applications. You need: Clarion C6.1 or higher; Microsoft .Net FrameWork 1.1 or higher. This package includes Clarion/.NET assembly DLL, Classes and Templates, demo app and documentation. Includes PropertyGrid control, which displays properties for any object or type, and it retrieves the item's properties. Price: $40 (1 developer license).
Posted Tuesday, July 25, 2006

Image to PDF DLL

Utility Warrior has released a Clarion interface to its Image to PDF Dynamic Link Library (DLL). Image to PDF will convert one or more images (JPEG, TIFF, PNG, GIF, BMP, WMF, EMF, PCX and TGA) into a PDF document. Supports multi-image TIFFs, animated GIFs, JPEG image re-compression (if required) and the ability to auto-rotate images so that they are all in portrait or landscape orientation. You can add clickable image stamps (e.g. company logo to web site), passwords, bookmarks and can create simple full screen PDF slideshows - plus all of the usual PDF options. Free to download and try and no details are required in order to access the ZIP file.
Posted Thursday, July 06, 2006

Clarion Magazine

Global Variables, Threads, Critical interSections and the Dangers of Unprotected Sets, Part 2

by John Morter

Published 2006-06-29

In Part 1 of this article I wrote about providing safe access to Global variables in Clarion's new(ish) pre-emptive thread model environment, using critical (inter)sections.

I've decided that I like critical (inter)sections, for a number of reasons; i) They're quick and simple to implement, ii) they're quite efficient (I have proven that, to my own satisfaction, in timing tests) and iii) they're very flexible (one can invoke them at will).

However, there's also a potential and serious problem involved. Each `Wait`(for a green light) that's issued *must* be matched with a subsequent (It's OK now to)`Release`(the green light). Otherwise it's possible to cause a gridlock, as the result of one thread "hogging" the green light and preventing all other (rule abiding) threads from getting their turn through the critical intersection.

And preventing this problem is not as simple as it may first seem. The matching requirement is "logical", not just "structural". That is, it applies in the logical execution path. So, avoiding it is not as straightforward as having an equal number of `Release`(s) in your code as you have `Wait`(s). Rather, your code must *execute* an equal number of `Release`(s) as it does `Wait`(s).

I decided to implement a solution that allowed me to exploit all the "good things" about Critical interSections, whilst helping me to minimize the risk of "bad things" happening.

These were my solution design must-haves:

- I wanted an ABC-like implementation, so that my critical intersection management system would be easy to use, with all the gory logic hidden within a class.
- I didn't want to have to rethink the implementation of my critical intersection management system each time I used it, and I wanted its implementation to be so simple that I would happily use it all the time. In other words, I needed to wrap my class up in a template, for automatic implementation.
- I wanted some additional value added to the class; I'd been recently caught out with a bug that resulted from attempting to assign a string value to a Global too long for the Global to hold, and which was not immediately obvious because the definition of the Global was out-of-sight and long forgotten. I wanted to catch basic problems like this at development time.

- The central purpose of my class was to warn me if I had any outstanding Wait(s) at critical points in my application (such as, when exiting from a procedure) ... along with provision of some easy-to-use methods for safe and protected access to Global variables.

I ended up with the following methods in my Globals Protection (GP) class.

SetGlobal method

```
GP.SetGlobal PROCEDURE (*? GlobalVariable,<? LocalVariable>,|
                        BYTE pDoChecks=True)
  CODE
  SELF.Wait()
  IF pDoChecks ! Note: Checks Will be ASSERTed in Debug-mode ONLY
?   IF ~OMITTED(LocalVariable)
?     ASSERT(ISSTRING(GlobalVariable)=ISSTRING(LocalVariable),|
?      '<10>GP.SetGlobal<10>Warning: Variable-type mismatch in '|
?      & ' GP.SetGlobal <10>Attempting to Set: [<39>' |
?      & LocalVariable & '<39>]<10>')
?     IF ISSTRING(GlobalVariable)
?       ASSERT(LEN(CLIP(LocalVariable))<=LEN(GlobalVariable),|
?        '<10>GP.SetGlobal<10>Warning: GlobalVariable ' |
?        & '[SIZE(' & LEN(GlobalVariable) & ')] will receive '|
?        & 'truncation from [SIZE(' & LEN(LocalVariable) |
?        & ') = <39>' & CLIP(LocalVariable) & '<39>]<10>')
?      END
?    END
?
  END
  GlobalVariable = LocalVariable
  SELF.Release()
  RETURN
```

Comments and explanations:

- This method is used like this: `GP.SetGlobal(g:VariableName, LocalVariable)` or: `GP.SetGlobal(g:VariableName, 'String')` or `GP.SetGlobal(g:VariableName, 99)`.
- It's equivalent to assigning, say, `g:VariableName = LocalVariable` - but in a safe and caring way, as provided for by the matching `Wait()` and `Release()`.
- The Global variable name is passed into this class method by address (as specified by the '*' in the procedure prototype), thereby ensuring that nothing is actually done to or with the variable until it can be protected by the Wait().
- The "Local Variable" can be a `VariableName`, or a value (eg. `'Somestring'` or `256`) *or an expression*.

Therefore, this provides a safe way of assigning, say;

```
GP.SetGlobal(g:VariableName, (LocalBYTE * SIZE(LocalSTRING)))
```

with one proviso: it's advisable not to use global variables in the expression, because the passed expression will be evaluated before the `GP.SetGlobal` method executes, and then a copy of the result is passed in for assignment to the global variable. If variables used in that expression are subject to change by another thread, you could get some unexpected results. But typically you'll be passing constant values to the method, and assigning those values or the result of an expression containing known values to another variable, so that shouldn't be a limitation.

- The "Local Variable" can be omitted, in which case the result is equivalent to `g:VariableName = ''` or `g:VariableName = 0` depending on the Global variable's data-type (String or Numeric, respectively).
- There are checks to warn if a mixed data-type assignment is being used, such as `g:GlobalLONG = LocalSTRING`, or if an incoming string is too long for the Global variable to hold (for protection against those simple-but-hard-to-find bugs).

If, however, you wish to allow a mixed data-type assignment (say, to set the contents of a Global `LONG` to the value of a string known to contain numbers) then an optional third parameter, when set to False/zero, will cause the checks to be skipped. For example: `GP.SetGlobal(g:GlobalLONG, LocalNumericString, 0)`

- These checks will be ASSERTed only if your application has been compiled in Debug-mode (as defined in the application's Global Options). That is, only when you're developing and testing. The `?` characters in column one of these ASSERTions ensure that the conditions are not tested once your application has been compiled in Release mode.
- There's a guaranteed `Release()` for the `Wait()`. There's no threat of causing a thread "grid lock" with this method.

GetGlobal method

```
GP.GetGlobal PROCEDURE (*? LocalVariable, *? GlobalVariable,|
                        BYTE pDoChecks=True)
  CODE
  SELF.Wait()
  IF pDoChecks ! Note: Checks Will be ASSERTed in Debug-mode ONLY
?   ASSERT(ISSTRING(GlobalVariable)=ISSTRING(LocalVariable),|
?     '<10>GP.GetGlobal<10>Warning: Variable-type mismatch in '|
?     & GP.GetGlobal <10>Attempting to Get: [<39>' |
?     & GlobalVariable & '<39>]<10>')
?   IF ISSTRING(LocalVariable)
?     ASSERT(LEN(CLIP(GlobalVariable))<=LEN(LocalVariable),|
?      '<10>GP.GetGlobal<10>Warning: LocalVariable [SIZE(' |
?      & LEN(LocalVariable) & ')] will receive truncation from [SIZE(' |
?      & LEN(GlobalVariable) & ') = <39>' |
?      & CLIP(GlobalVariable) & '<39>]<10>')
?   END
  END
  LocalVariable = GlobalVariable
  SELF.Release()
  RETURN
```

Comments and explanations:

- This method is simply the opposite of `GP.SetGlobal`. It's used like this; `GP.GetGlobal(LocalVariable, g:VariableName)` - that is, it's the equivalent of `LocalVariable = g:VariableName`.
- As for `GP.SetGlobal`, the assignment process is protected by a `Wait()` and a matching `Release()`. Therefore, this method provides an entirely safe way to involve Global variables in an expression, without concern about the potential for a thread-switch part way through the calculation of any intermediate steps. For example;

```
GP.GetGlobal(LocalVariable, (g:GlobalBYTE * SIZE(g:GlobalSTRING))) .
```

- All else is pretty much the same as for `GP.SetGlobal` except, of course, that the global variable name/value/expression cannot be omitted.

Testing

I am now equipped to repeat my earlier tests and check the results:

```
PartiallySafeTestA   PROCEDURE


LocalLONG   LONG
LocalString STRING('AAAAAAAAAA')
PEEKString  STRING(50)


  CODE

  RESUME(START(PartiallySafeTestB))
  ! START new thread w/o waiting for this one to complete

  LOOP i# = 1 TO 100000
    GP.SetGlobal(GlobalLONG,(LocalLONG+1))

    GP.SetGlobal(GlobalSTRING,(LocalString & LocalString & |
      LocalString & LocalString & LocalString))
    PEEK(ADDRESS(GlobalSTRING),PEEKString)
    IF PEEKString <> ALL('A',50) AND PEEKString <> ALL('B',50)
      STOP('PartSafeA: LocalString='& LocalString |
         &' Thread#='& THREAD() |
         &'  i#='& i# &' LocalLONG = ' & LocalLONG |
         &'<10>Ooops !<10>GlobalString = '& PEEKString)
      BREAK
    END
    GP.GetGlobal(LocalLONG,GlobalLONG)
  END
  STOP('PartSafeA: LocalString='& LocalString &' Thread#='& THREAD() |
      &'  i#='& i# &' LocalLONG = ' & LocalLONG)

!-----------------------------------------------------------------
```

```
PartiallySafeTestB   PROCEDURE

LocalLONG   LONG
LocalString STRING('BBBBBBBBBB')
PEEKString  STRING(50)
  CODE
  LOOP i# = 1 TO 100000
    GP.SetGlobal(GlobalLONG,(LocalLONG+1))
    GP.SetGlobal(GlobalSTRING,(LocalString & LocalString |
       & LocalString & LocalString & LocalString))
    GP.GetGlobal(LocalLONG,GlobalLONG)
  END
  STOP('PartSafeB: LocalString='& LocalString &' Thread#='& THREAD() |
      &'  i#='& i# &' LocalLONG = ' & LocalLONG)
```

I was initially perplexed by the result that I got, in Figure 1.

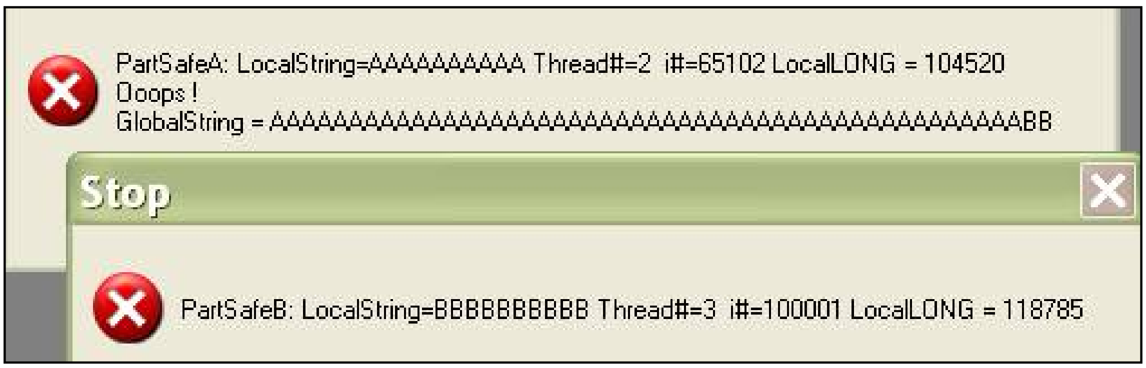


Figure 1. A false alarm

But then I realized it was my testing that was at fault.

What's happening is that, although the individual Global variable assignments (via the sets and gets) are safe, this safety (afforded by the `Wait`(s) and `Release`(s) in which they're encapsulated) does not apply "between" the lines of code in these procedures. And, although it took quite a few iterations to trip-up (65,102 in this example), eventually a thread-switch occurs at exactly the wrong time... and Ooops! is the result.

You'll need to trust me here (or test this to your own satisfaction); I temporarily moved the test-for-Ooops code to inside the `GP.SetGlobal` method, and no Ooopses were detected.

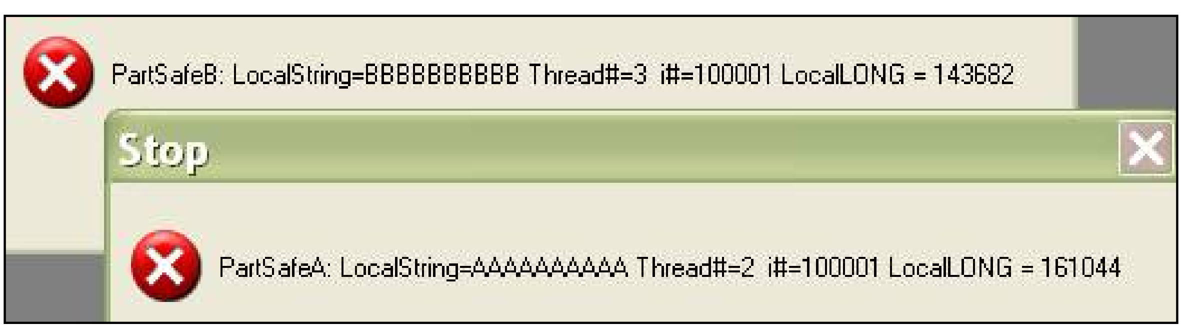


Figure 2. That's more like it ... Sort of !

However, there's a lesson here – and the "interesting" result in `LocalLONG` (also noticed early in Part 1) lends weight to this lesson.

In case the "strangeness" about the result in `LocalLONG` is not immediately obvious, I'll explain: Take a closer look at the relevant code:

```
GP.SetGlobal(GlobalLONG,(LocalLONG+1))!eq. to GlobalLong = LocalLONG + 1
GP.GetGlobal(LocalLONG,GlobalLONG)    !eq. to LocalLONG = GlobalLONG
```

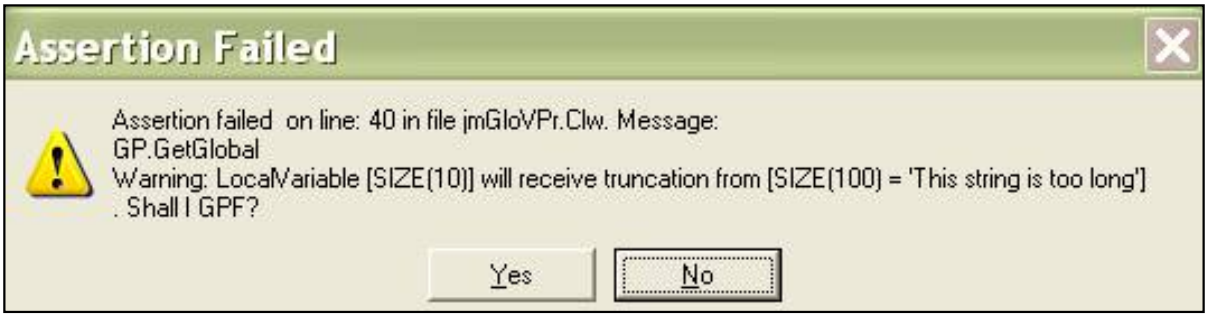
In the actual test-code I've separated these lines with the `GlobalSTRING` assignment, but that was just to allow more time for something to go wrong.

One's first-glance expectation for the result in `LocalLONG` is the same as for the result in the implicit numeric variable, `i#` (which is declared *locally*, and is therefore naturally protected). However, by handing the value in `LocalLONG` over to `GlobalLONG`, albeit very temporarily, opportunity is set up for the other thread to grab its turn between the time of the handover and the subsequent retrieval. Eventually, the two threads trip over each other and the resulting values in both instances of `LocalLONG` spin way out of expected range.

The lesson is this: If your work with global variables extends over multiple lines of code then it's not good enough to protect individual assignment statements from interruption by a thread-switch ... you must protect the entire passage of execution, until some point where you're comfortable for (the potential of) a thread-switch to occur.

This leads me naturally to a description of the other methods in my Globals Protection class. However, before I go there, I'll deal with the question I asked myself at this stage: "Is there any point then in bothering with the GP.SetGlobal/GetGlobal methods?".

After a bit of thought, I decided the answer is "Yes", because there are many times when all that's required is to assign or retrieve a value from a global variable , and these methods provide a totally safe and very easy-to-use means of doing so (in minimal lines of code). Besides, they also provide the added value of automatic checking for "silly" coding mistakes (but only during development & testing stage, when my application is compiled in debug-mode).



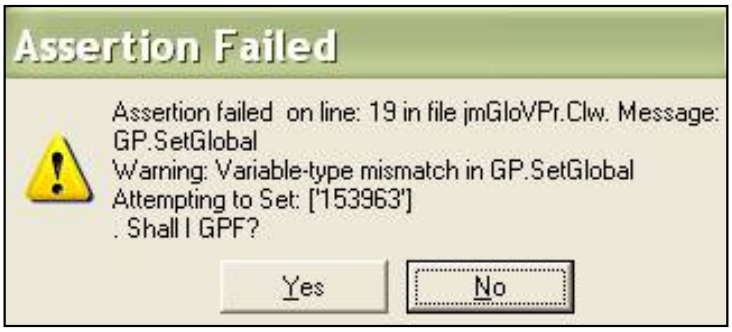


Figure 3. Silly me!!

Now, on to a description of the remaining methods in my Globals Protection class;

```
GP.WaitForGreen      PROCEDURE
  CODE
  SELF.Wait()        ! Calls to Wait() can be nested
  GP.TotWaitCount += 1
  GPt:WaitCount   += 1
  RETURN
```

If you've been following my "Traffic Cop" analogy then this will be perfectly obvious to you. `GP.WaitForGreen` is pretty much just a `Wait()` in disguise, but with some additional "smarts" that will make better sense very shortly.

It's important to note here that `GPt:WaitCount` is a `THREAD`ed global variable; it's keeping track of the number of outstanding `Wait`(s) for each thread. Conversely, `GP.TotWaitCount` is not `THREAD`ed; it's keeping track of the overall number of outstanding `Wait`(s).

Because I'm careful to work with `GP.TotWaitCount` only *within* the safety of a `Wait()/Release()`, there is no risk of two or more threads mucking around with this counter at the same time.

```
GP.ReleaseGreen      PROCEDURE
  CODE
  IF GPt:WaitCount
   ! Number of calls to Release() MUST NOT exceed calls to Wait()
    GP.TotWaitCount -= 1
    GPt:WaitCount   -= 1
    SELF.Release()
  ELSE
?   ASSERT(0,'<10>GP.ReleaseGreen<10>Wait/Release mismatch: '|
?      & 'Call made to GP.ReleaseGreen without preceding '|
?      & 'call to GP.WaitForGreen<10>')
  END
  RETURN
```

Before the `Release()` is issued, a check is made to ensure there truly is a `Wait()` outstanding *for the current thread*. If so, the `Release()` proceeds and the count of outstanding `Wait`(s) can be decremented.

Otherwise, provided the application is compiled in Debug mode, an `ASSERT`ion failure is returned to warn

about the `Wait/Release` mismatch encountered in the execution path.

I am now at another point where I can run some tests and check the results.

```
FullySafeTestA       PROCEDURE
LocalLONG   LONG
LocalString STRING('AAAAAAAAAA')
PEEKString  STRING(50)
  CODE
  RESUME(START(FullySafeTestB))
  ! START new thread w/o waiting for this one to complete

  LOOP i# = 1 TO 100000
    GP.WaitForGreen !<<<
    GlobalLONG = (LocalLONG+1)
    GlobalSTRING = (LocalString & LocalString & LocalString & |
      LocalString & LocalString)
    PEEK(ADDRESS(GlobalSTRING),PEEKString)
    IF PEEKString <> ALL('A',50) AND PEEKString <> ALL('B',50)
      STOP('FullySafeA: LocalString='& LocalString |
         &' Thread#='& THREAD() |
         &'  i#='& i# &' LocalLONG = ' & LocalLONG |
         &'<10>Ooops !<10>GlobalString = '& PEEKString)
    END
    LocalLONG = GlobalLONG
    GP.ReleaseGreen !<<<
  END
  STOP('FullySafeA: LocalString='& LocalString &' Thread#='|
    & THREAD() &'  i#='& i# &' LocalLONG = ' & LocalLONG)

!---------------------------------------------------------------

FullySafeTestB       PROCEDURE
LocalLONG   LONG
LocalString STRING('BBBBBBBBBB')
PEEKString  STRING(50)
  CODE
  LOOP i# = 1 TO 100000
    GP.WaitForGreen !<<<
    GlobalLONG = (LocalLONG+1)
    GlobalSTRING = (LocalString & LocalString & LocalString & |
      LocalString & LocalString)
    LocalLONG = GlobalLONG
    GP.ReleaseGreen !<<<
  END
  STOP('FullySafeB: LocalString='& LocalString &' Thread#='|
    & THREAD() &'  i#='& i# &' LocalLONG = ' & LocalLONG)
  GP.ReleaseGreen !<<< Nothing to Release(!)
```

I’m very happy with this result ...

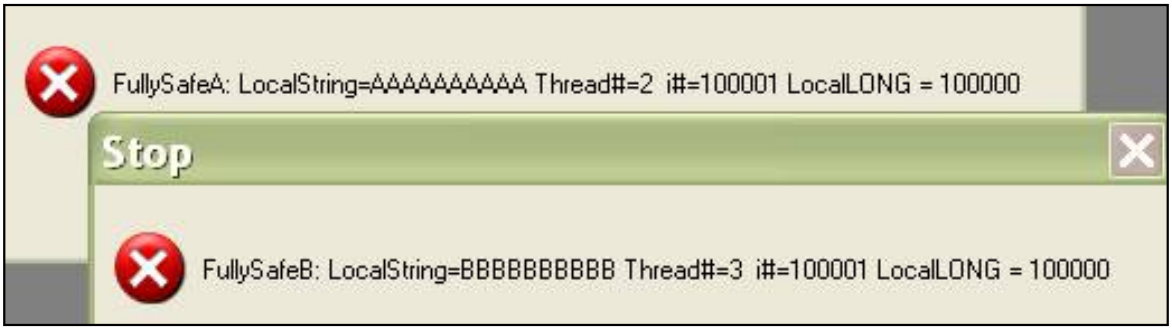


Figure 4. Looking good – No "funny stuff" here.

And did you notice the unmatched `GP.ReleaseGreen` at the end of `FullySafeTestB` ?

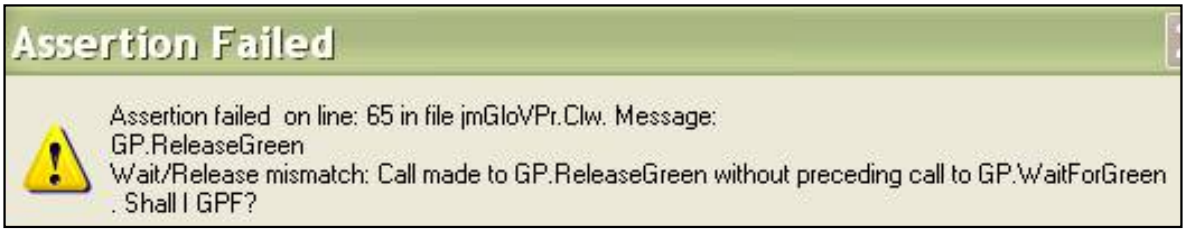


Figure 5. Gotcha !

And, the point of the overall count of outstanding `Wait`(s)?

As explained by Clarion Help:

> A CLASS method labeled "Destruct" is a method which is automatically invoked when the object leaves scope.

That is, when an application terminates, just before the class/object and its associated memory contents are destroyed, any method named `Destruct` is automatically called.

My `GP` class has a `Destruct` method that `REPLACE`s the `Destruct` method defined for the class on which my `GP` class is actually based (that is, SoftVelocity's `CriticalSection` class). This allows for some additional checks to be made during the last gasps of the application.

```
GP.Destruct          PROCEDURE
  CODE
  IF GP.TotWaitCount   ! Not ALL Wait(s) have been Release(d)
    LOOP UNTIL GP.TotWaitCount = 0
      SELF.Release()
      GP.TotWaitCount -=1
    END!LOOP
?   ASSERT(0,'<10>GP.Destruct<10>Wait/Release mismatch: '|
?      & 'TOTAL number of calls to GP.WaitForGreen exceeded '|
?      & 'the TOTAL number of calls to GP.ReleaseGreen<10>')
  END
  ! Now do whatever SV's CriticalSection.Destruct normally does
  PARENT.Destruct
```

```
    RETURN
```

A Template wrapper

An associated template makes the Class very simple to implement, and it provides the option to add even more rigorous checking for `Wait()/Release()` mismatches.

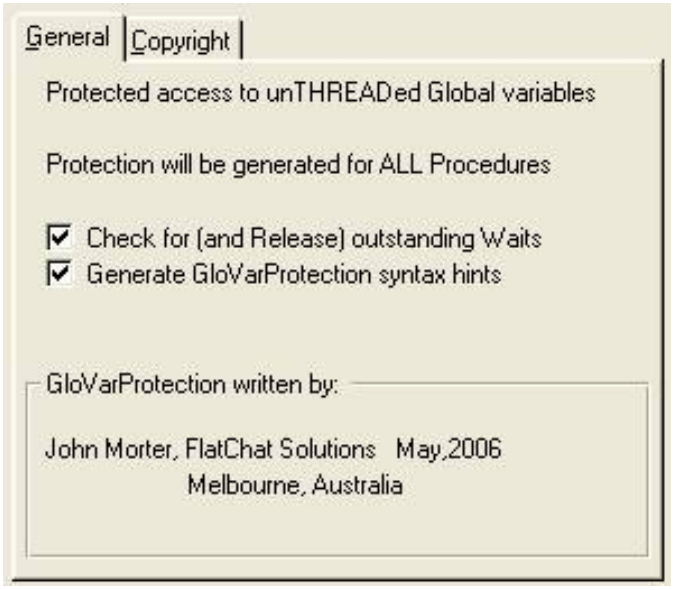


Figure 6. Global (Application-level) Extension Template

Simply add this extension template (via your application's Global Properties) for automatic instantiation of the Globals Protection (GP) class, and you may then immediately use all the GP methods, as detailed above, anywhere in your embed code.

By checking (or ticking, depending upon your cultural background) the option to Generate GloVarProtection syntax hints, you will cause the template to insert some reminders about usage of the `GP.Methods` at the end of each application procedure.

The other option, to Check for (and Release) outstanding Waits, generates a test similar to the one described above that's triggered when the class `DESTRUCT`s, but this one is fired at the exit point of each application procedure and it checks only for `Wait`(s) outstanding for the current thread. The bold line of code below results from checking/ticking this option.

```
ThisWindow.Kill PROCEDURE

ReturnValue          BYTE,AUTO

  CODE
  ReturnValue = PARENT.Kill()
  IF ReturnValue THEN RETURN ReturnValue.
  IF SELF.FilesOpened
    Relate:DummyTable.Close
  END
```

```
    IF SELF.Opened
      INIMgr.Update('DummyProcess',ProgressWindow)
    END
    GlobalErrors.SetProcedureName
    GP.MatchWaitRelease(GP:ActionFlag) !Generated by GloVarProtection template
    RETURN ReturnValue
```

Note use of the `ActionFlag` parameter to enable customization of the action to be taken when a mismatch is detected.

```
GP.MatchWaitRelease  PROCEDURE (BYTE pActionFlag=1)
RetVal               BYTE(Level:Benign)
  CODE
  IF GPt:WaitCount
    ! Not all Wait(s) for this Thread have been Release(d)
    RetVal = Level:Notify
    IF pActionFlag = 1 OR pActionFlag = 2
?     ASSERT(0,'<10>GP.MatchWaitRelease<10>Wait/Release mismatch: '|
?        & 'Number of calls to GP.WaitForGreen exceeds the number '|
?        & of calls to GP.ReleaseGreen<10>')
    END
    IF pActionFlag = 2 OR pActionFlag = 3
      LOOP UNTIL GPt:WaitCount = 0
        ! Release all outstanding Wait(s) for this Thread
        SELF.ReleaseGreen
      END!LOOP
    END
  END
  RETURN RetVal
```

The default setting (= 1) for `GP:ActionFlag` can be overridden in Embed code.

If any outstanding `Wait`(s) are detected, then;

- A value of 2 results in display of the ASSERTion message *and* the `Release()` of all outstanding `Wait`(s) *for the current thread*.
- A value of 1 results in display of the ASSERTion message, only.
- A value of 3 results in the `Release()` of all outstanding `Wait`(s), only.
- A value of `0/False` causes any outstanding `Release`(s) to be completely ignored.

I am finding it useful to call the `GP.MatchWaitRelease()` method from Embed points just below where I have conditional calls to `WaitForGreen()` and `ReleaseGreen()` to ensure I have them properly matched in the logical execution path. I usually allow the default value for `ActionFlag (=1)` to apply, because then I have the comfort of knowing that the ASSERTion check will always be performed for me whilst I'm testing (with my application compiled in Debug mode) and that it will be ignored once my application has been shipped (after being thoroughly tested and re-compiled in Release mode).

Putting it all together

Attached to this article you will find four files:

- jmGloVPr.inc and jmGloVPr.clw contain the Globals Protection (GP) class definitions and logic. They should be placed in your %Root%\LibSrc folder.
- AB_AddOn.tpl and jmGloVPr.tpw are the template files. They should be placed in your %Root%\Template folder.
- AB_AddOn.tpl is a "container" for templates I collect and write for myself. To make available to your application the templates defined therein, you will need to register this template-chain via the Setup|Template Registry menu option.

I do hope you will find this class and template set useful. At very least, my aim has been to share my discoveries about critical (inter)sections with the Clarion community (at the same time as earning some more Clarion Magazine subscription credits!).

Acknowledgements

I used CapeSoft's Object Writer templates to create my Class files (jmGloVPr.inc and jmGloVPr.clw). The documentation that comes with this freebie from CapeSoft provides a wealth of good information about creating and using classes, and the value-for-money is unbeatable.

At one point, whilst grappling with some problems I was having with variables/properties in my class, I put out a request for help on the softvelocity.public.clarion6 newsgroup, and I received an almost instant response from Jim Kane. Wow! Guidance from "The Master" (thanks Jim).

Download the source

John Morter is a member of the Victorian Clarion Users Group (Melbourne, Australia). John is Asia Pacific IT Manager for a brand-name multi-national and he's supposed to leave all the fun technical stuff for others to do. So, his Clarion work is developed under the nom-de-keyboard Flat Chat Solutions, where "flat chat" is an Australian expression meaning doing something at top speed / high velocity.

Reader Comments

Add a comment

Clarion Magazine

Global Variables, Threads, Critical interSections and the Dangers of Unprotected Sets

by John Morter

Published 2006-06-22

The recent Aussie DevCon was an excellent event, and we were most fortunate to have the knowledgeable and entertaining Bruce Johnson and Jono Woodhouse (a.k.a. "The Capesoft Clowns") as presenters, especially when they delved into some of the meatier topics, such as C6 threading.

Bruce started out with an explanation of some of the pitfalls associated with the changes delivered with Clarion 6 to support the pre-emptive thread model. You can read more about Bruce and Jono's presentation in Geoff Robinson's report.

For those of you wondering whether all the fuss and angst heaped upon us by this thread-management de-simplification is worth all the trouble, Bruce gave one example that "sold" me: It's Clarion's new threading model capability that enables us to write Windows Service applications. Essentially, it's another modernisation of our development language of choice.

Looking back to what I'd gleaned for myself, up until when Bruce & Jono put me straight, I now realize that I knew just enough to be dangerous: I had a few misconceptions, as you will see.

I'm not going to attempt to explain the change introduced with C6 (from a cooperative threading model), other than to recommend you to the SoftVelocity "Multi-Threaded Programming Guide". It may take multiple reads to make any sense out of it (at least, that was my experience), but every time you refer back to it you're sure to hear another penny drop! There have also been a number of excellent Clarion Magazine articles on this topic. These two resources reinforce each other well.

Common misconceptions

I knew, from my Clarion Magazine reading, that dangers lurked when assigning values to global `STRING`s because it's possible for a thread-switch to occur mid-way through the assignment, (such that the global variable ends up containing part of what one thread was assigning to it and part of what another thread was half-way through doing before it was interrupted). But I thought I was safe assigning values to `LONG`s and `BYTE`s, etc., because assignment to these variables occurs at the "atomic level" (in a single Assembler-level instruction). However, Bruce pointed out a subtlety that's now oh-so-obvious.

```
g:GlobalLONG = g:GlobalLONG * (g:GlobalBYTE + LocalBYTE)
```

This is an example that could get you into trouble because, although it's true that the final assignment to `g:GlobalLong` will occur within one instruction, you cannot be sure what intermediate steps the compiler may generate, and it's possible that a fateful thread-switch could change the value of any of those intermediaries – with a scrambled result.

I decided to do some investigation into these possibilities, with the following code:

```
!================================================================
TotallyUnsafeTestA   PROCEDURE
LocalLONG   LONG
LocalString STRING('AAAAAAAAAA')
PEEKString  STRING(50)
  CODE
  RESUME(START(TotallyUnSafeTestB))
! START new Thread w/o waiting for this one to complete
!
  LOOP i# = 1 TO 100000
    GlobalLONG = LocalLONG+1
!
    GlobalSTRING = LocalString & LocalString & LocalString & |
                   LocalString & LocalString
    PEEK(ADDRESS(GlobalSTRING),PEEKString)
    IF PEEKString <> ALL('A',50) AND PEEKString <> ALL('B',50)|
      STOP('UnsafeA: LocalString='& LocalString &' Thread#='& |
           THREAD() &'  i#='& i# &' LocalLONG = ' & LocalLONG |
                &'<10>Ooops !<10>GlobalString = '& PEEKString)
      BREAK
    END
    LocalLONG = GlobalLONG
  END!Loop
!----------------------------------------------------------------
TotallyUnsafeTestB   PROCEDURE
LocalLONG LONG
LocalString STRING('BBBBBBBBBB')
  CODE
  LOOP i# = 1 TO 100000
    GlobalLONG = LocalLONG+1
    GlobalSTRING = LocalString & LocalString & LocalString & |
                   LocalString & LocalString
!
    LocalLONG = GlobalLONG
  END!Loop
  STOP('UnsafeB: LocalString='& LocalString &' Thread#='& |
       THREAD() &'  i#='& i# &' LocalLONG = ' & LocalLONG)
!================================================================
```

The first procedure (`TotallyUnsafeTestA`) invokes the second (`TotallyUnsafeTestB`) and they both update global data, just inviting something to go wrong … and it soon does.

Note: Global variables are defined as `GlobalSTRING STRING(100)` and `GlobalLONG LONG`.

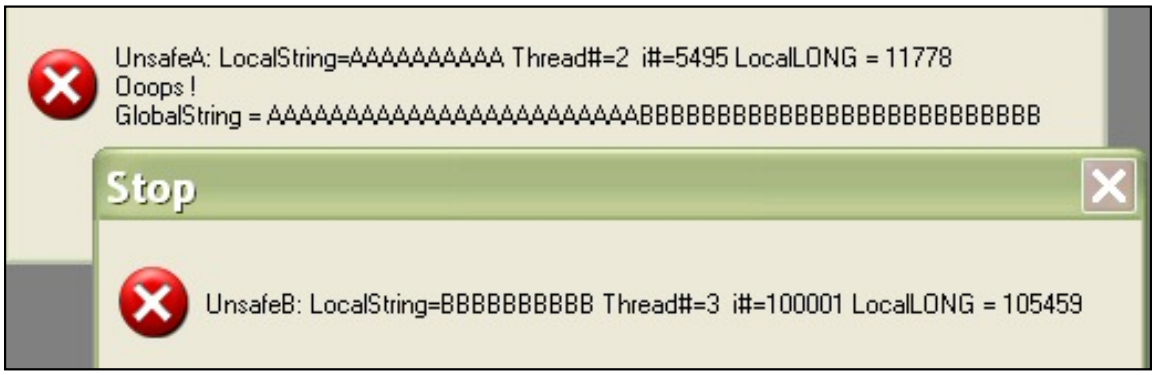


Figure 1. The danger of unprotected sets

After 5,495 loops through this test (I've seen this as low as 17!) the two threads have tripped over each other, and the result in `GlobalSTRING` is certainly not what one might expect.

Note that there's also something "interesting" going on with `LocalLONG`... I'll come back to that later (don't let me forget).

In everyday practice, with the typical user-driven application, this problem is unlikely to occur very often(!), but it's the clearly proven fact that it can occur that should make us all much more careful when working with global variables.

And, it actually gets worse!

From bad to worse

I didn't understand what the problem might be with global queues, provided they are primed at program start-up (when there is only one thread) and then only ever read, never updated, by subsequent application threads. However, as Bruce explained, a pointer keeps track of the "current position" in the queue. And this pointer is a global variable (unless the queue is THREADed, which would defeat the purpose of a shared queue). So, it would probably not take too long for two threads, reading through the same queue, to get themselves confused.

The clear and simple message imparted by Bruce and Jono was; *"Don't use global variables if you can avoid doing so … but, if you must, then use some guaranteed protection"*.

The lessons I learned from my investigations for this project have certainly dissuaded me from using global variables in any new application I write. But most of us have older apps around; with "old-fashioned" programming techniques built-in, and great care is needed when converting these to C6, and beyond.

How is this done?

Firstly, it is safe to use variables as global constants [eg. `GlobalConstant STRING('Clarion 6.3')`], provided they are never (that's never, ever) changed within the application (unless you use some of the protection techniques explained later in this article, which is where I'm slowly heading).

However, the potential problem is that while you may have had no intention of changing a variable's value when you originally wrote the application, some future requirement may see you (unthinkingly) updating it without realizing the dangers.

A naming convention is a good way to remind yourself about the state and purpose of your application variables; especially for global variables, because their definition is usually well out-of-sight when you're coding those potentially dangerous embeds.

For example:

```
gt:ThreadedGlobal  LONG,THREAD        !Thread-safe
gc:GlobalConstant       STRING('Colour')   !Must never be updated
```

Better yet:

```
ge:GlobalEquate    EQUATE('Hue')       !Never CAN be updated
```

As Jono pointed out, changing the names of your global variables is an excellent way to flush-out all those places where you may have misused them … because the compiler will very quickly identify them all for you!

Many of us have fallen into bad habits with regard to using global variables as a means of communicating between procedures, whereas the safe (and more professional) approach is to pass parameters to procedures and receive return values from them. For a working example of the syntax involved, see the `CityStateZip` procedure in the Invoice.APP sample application.

All the same, there will probably be times when use of a global variable is the most pragmatic solution to a programming requirement, but we must guard against the dangers involved.

Practicing safe sets

It's obvious that we need a way to ensure, when we set a value to a global variable, that we end up with a guaranteed and expected result, because being correct "most times" is not good enough for most of our users.

The solution recommended by Bruce & Jono was straightforward; use *critical sections* to protect access to your global variables, and use them consistently. This makes good sense. It's not wise to have conditional approaches to doing similar things, because you're likely to make the occasional incorrect decision about which alternative you apply.

The ubiquitous Steve Parker published an article titled Implementing a Critical Section: Fast and Effective that explains how to implement global variable protection in just four lines of code.

I decided to test the theory:

```
!============================================================
TotallyUnsafeTestC   PROCEDURE
LocalLONG   LONG
```

```
LocalString STRING('AAAAAAAAAA')
PEEKString  STRING(50)
  CODE
  RESUME(START(CriticalSectionTestB))
! START new thread w/o waiting for this one to complete
!
  LOOP i# = 1 TO 100000
    GlobalLONG = LocalLONG+1
!
    GlobalSTRING = LocalString & LocalString |
      & LocalString & LocalString & LocalString
    PEEK(ADDRESS(GlobalSTRING),PEEKString)
    IF PEEKString <> ALL('A',50) AND PEEKString <> ALL('B',50)
      STOP('UnsafeC: LocalString='& LocalString |
          &' Thread#='& THREAD() |
          &'  i#='& i# &' LocalLONG = ' & LocalLONG |
          &'<10>Ooops !<10>GlobalString = '& PEEKString)
      BREAK
    END
    LocalLONG = GlobalLONG
  END!Loop
!-----------------------------------------------------------------
CriticalSectionTestB PROCEDURE
LocalLONG LONG
LocalString STRING('BBBBBBBBBB')
  CODE
  LOOP i# = 1 TO 100000

    GP.Wait()
    GlobalLONG = LocalLONG+1
    GlobalSTRING = LocalString & LocalString & |
      LocalString & LocalString & LocalString
    LocalLONG = GlobalLONG
    GP.Release()

  END!Loop
  STOP('CriticalSectionB: LocalString='& |
     LocalString &' Thread#='& THREAD() |
     &'  i#='& i# &' LocalLONG = ' & LocalLONG)
!=================================================================
```

I'm using the same testing technique as shown in Figure 1, above. This time, however, I've used a critical section to protect the global variables … or so I thought. (See Steve Parker's article for the details. My "`GP`" prefix is equivalent to Steve's "`MyCriticalSection`" prefix.)

Figure 2 shows the results:

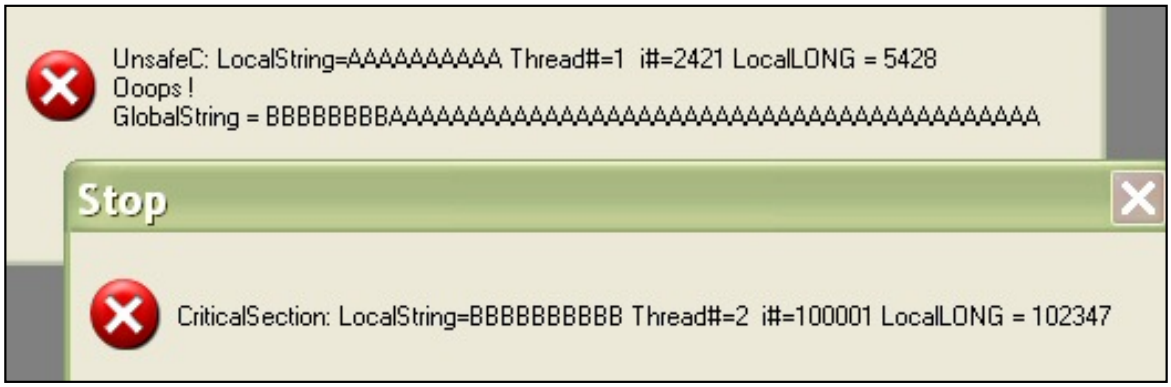


Figure 2. The danger of a little bit of knowledge

Mmmmmm… Not quite what I was expecting!

You will recall I mentioned some initial misconceptions. You see, I was accepting what I was reading much too literally. I was (mis)understanding that a critical section would "lock" my globals so that other threads couldn't corrupt them. Assuming that `Wait()` meant "wait for a lock" and `Release()` meant "now release the lock" seemed quite reasonable to me back then. But that wrongly assumed that Lock and Release were doing something unseen to my variables.

OK. Testing involves trying out different approaches and reviewing the results. After pursuing a couple of other dead-ends, I ended up with this next example.

```
!================================================================
CriticalSectionTestA PROCEDURE
LocalLONG   LONG
LocalString STRING('AAAAAAAAAA')
PEEKString  STRING(50)
  CODE
  RESUME(START(CriticalSectionTestB))
! START new thread w/o waiting for this one to complete
!
  LOOP i# = 1 TO 100000
    GP.Wait()

    GlobalLONG = LocalLONG+1
    GlobalSTRING = LocalString & LocalString |
      & LocalString & LocalString & LocalString
    PEEK(ADDRESS(GlobalSTRING),PEEKString)
    IF PEEKString <> ALL('A',50) AND PEEKString <> ALL('B',50)
      STOP('CriticalSectionTestA: LocalString='|
        & LocalString &' Thread#='& THREAD() |
        &'  i#='& i# &' LocalLONG = ' & LocalLONG |
        &'<10>Ooops !<10>GlobalString = '& PEEKString)
      GP.Release()
      BREAK
    .
    LocalLONG = GlobalLONG
    GP.Release()
```

```
  END!Loop
  STOP('CriticalSectionA: LocalString='& LocalString &' Thread#='& THREAD() |
       &'  i#='& i# &' LocalLONG = ' & LocalLONG)
!---------------------------------------------------------------
CriticalSectionTestB PROCEDURE
LocalLONG LONG
LocalString STRING('BBBBBBBBBB')
  CODE
  LOOP i# = 1 TO 100000
    GP.Wait()

    GlobalLONG = LocalLONG+1
    GlobalSTRING = LocalString & LocalString |
      & LocalString & LocalString & LocalString
    LocalLONG = GlobalLONG
    GP.Release()

  END!Loop
  STOP('CriticalSectionB: LocalString='& LocalString |
    &' Thread#='& THREAD() |
    &'  i#='& i# &' LocalLONG = ' & LocalLONG)
!===============================================================
```

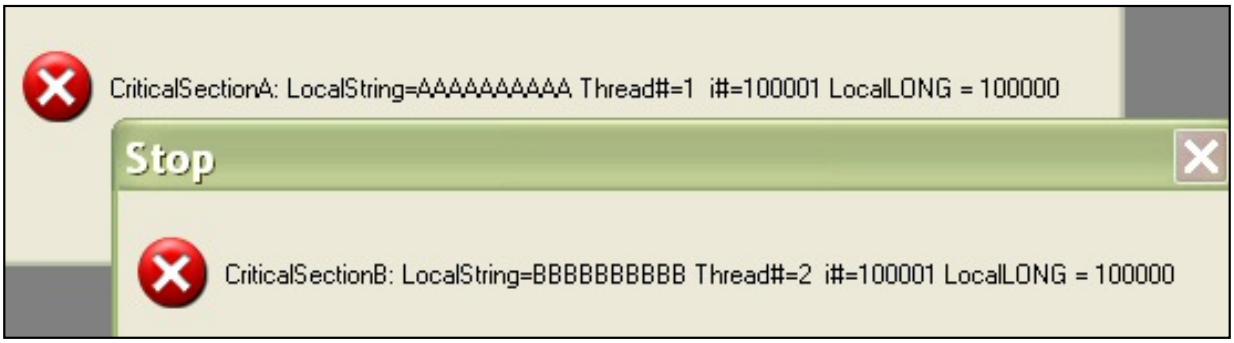


Figure 3. Success, at last.

I stopped and thought about this for a while … actually, for quite a while. Then I went to bed. Next morning the situation was much clearer to me. I now realize that a critical section is really more like a critical *inter*section.

It works like this: Each thread happily drives down its execution path until it reaches a critical intersection, marked by a `Wait()`. At this point it asks The Senior Traffic Cop (aka the Windows Operating System): "Is anyone else working in the critical intersection just now?". The STC checks her list and, when she's satisfied that no other thread is in the way, she gives the next thread in the queue a green light. When the thread has finished what it was doing in the middle of the critical intersection, and it has emerged safely out the other side, then it's good behaviour to tell The STC that it's safe to hand-over the green light to another thread (because, otherwise, none of the other threads will ever get their turn). This "OK, I'm finished" is communicated via a Release().

Now look back to the source code above Figure 2.

The `TotallyUnsafeTestC` thread was acting like one of those drivers you see in the news. He was speeding down his execution path, oblivious to all other threads on the road, and drove straight through the critical intersection without first checking that he had a green light … just as the other thread was bending over to pick

up that GlobalSTRING. The impact was fatal!

As is true on our highways and byways, threads (drivers) are not safe unless *everyone* is obeying the same road-rules. The Figure 3 source code demonstrates how to arrive safely.

OK, I'm comfortable with that. So all I need to do is make absolutely sure that I never (that's never, ever) forget to ensure that every foray into a critical intersection is made safely and consistently … by Wait(ing) for a green light first, and always Release(ing) the green light when I'm finished. Too easy!

And then I re-read Steve's article:

> Watch Out! There is only one caveat, one thing you *must* be careful about. If you do not release the critical section at the earliest possible moment, your application will appear to hang. Whenever another procedure tries to enter the critical section, if the previous one has not been released, it will be blocked until such time as the first procedure releases the critical section.

So, it's not quite so straightforward as simply matching each Wait() with a Release(), such as one does with IF/END constructs. Rather "logical" matching is required, so that each Wait() must be matched *in the execution path* by a Release().

```
CriticalSectionTestA PROCEDURE
LocalLONG    LONG
LocalString STRING('AAAAAAAAAA')
PEEKString   STRING(50)
  CODE
  RESUME(START(CriticalSectionTestB)) ! START new Thread w/o waiting for this one to complete
!
  LOOP i# = 1 TO 100000

    GP.Wait()
    GlobalLONG = LocalLONG+1
    GlobalSTRING = LocalString & LocalString & LocalString & LocalString & LocalString
    PEEK(ADDRESS(GlobalSTRING),PEEKString)
    IF PEEKString <> ALL('A',50) AND PEEKString <> ALL('B',50)
      STOP('CriticalSectionTestA: LocalString='& LocalString &' Thread#='& THREAD() |
            &'  i#='& i# &' LocalLONG = ' |
            & LocalLONG &'<10>Ooops !<10>GlobalString = '& PEEKString)
      GP.Release()
      BREAK
    .
    LocalLONG = GlobalLONG
    GP.Release()

  END
  STOP('CriticalSectionA: LocalString='& LocalString &' Thread#='& THREAD() |
      &'  i#='& i# &' LocalLONG = ' & LocalLONG)
```

Figure 4. "Logical" matching of Wait(s) and Release(s).

An example of this can be seen in Figure 4, where there are two Release() statements for the one Wait(). The Release() just above the BREAK is essential to ensure that The STC is advised that the critical intersection green light currently "owned" by that thread can be passed on to another thread. Otherwise, the circumstance that Steve warns us about will certainly occur.

Mmmmmm.... I'm not so comfortable with this any more.

I figured there had to be a better way of ensuring that global variables are always fully protected. I wasn't at all happy about a "solution" that fixed one problem only to introduce the potential of causing a different problem somewhere else.

For a while I thought I'd found the ideal solution when I read Steve's follow-on article Critical Procedures: Synchronisation for the Lazy. Steve writes:

> "The main advantage of using the `CriticalProcedure` class to handle the locking and releasing for you is that if you have multiple `RETURN` statements in your procedure.... This means that no matter how many Return statements I have in my procedure, all the exit points are covered."

However, this is *not* a good technique to use in most situations because there's no way to force the `Release()` of a `CriticalProcedure`, as Steve makes clear in his article. For example, if a `CriticalProcedure` is used in the `ThisWindow.Init` section of a browse (and assuming that all threads are "driving according to the rule-book") then any other thread will be stalled until the browse procedure has completed, which may not be until its user gets back from lunch!

Color tricks

Aside: If you're reading this in colour then you may have noticed that certain Clarion keywords in my code are highlighted in bright magenta. This is yet another nifty trick that I picked up from Jono Woodhouse.

Having the `Wait` and `Release` statements stand out so clearly is a very helpful way of checking, visually, that you have the necessary match-ups in place. Similarly, as Jono noted, it's handy being able to spot all those `STOP` statements that you included while debugging & testing, but forgot to remove from the shipping version. (How embarrassment! [sic])

The allocation of colours to Clarion keywords, as used by the Clarion source-code Editor, is made via the IDE menu (Setup > Editor Options > Colors … spelt "wrongly" !!). Assignment of Clarion keywords to `Color_Values` and the assignment of `Color_Values` to `Color_Groups` (to which the colours are actually assigned, via the aforementioned Editor Options > Colors dialogue) is made within CxxEDT.INI (eg. C60EDT.INI).

See Customizing Clarion5's Editor And Menus for a thorough discussion on how to fiddle-about with Clarion's configuration INI files. This article is quite old now, but it's not dated. Everything in it applies equally to all versions of Clarion since then.

There's gotta be a better way

The idea for a "Global Variables Protection Class" actually formed in my mind as I was listening to Bruce & Jono's presentation on the merits and perils of C6 threading. (Although, as demonstrated above, I didn't appreciate at all back then what a long way I had to go to properly understand what I was trying to achieve.)

In the next installment of this article I'll continue with my explanation of this learning process and of the resulting Class & Template set, which packages-up a number of approaches to ensuring protection of global variables.

(No, I haven't forgotten about the strange goings-on with that `LocalLONG`... I'll cover that in the next part too.)

Acknowledgements, thus far

Thanks, obviously, go to Bruce, Jono and Steve.

I'd hate Russ Eggen to think (when he sees all those `STOP`s in my code) that I wasn't listening to his Aussie DevCon presentation on using the Clarion Debugger. On the contrary, he motivated me to get back into the habit of using the Debugger, which I found to have improved in quite a few subtle ways since I last used it, and it was invaluable in my investigations and testing. (There's still a place for very careful use of `STOP`s too, but).

Thank you also to Phil Will at ProDomus for his generous sponsorship of the Aussie DevCon; I was lucky enough to win a selection of his Edit & Lookup tools.

John Morter is a member of the Victorian Clarion Users Group (Melbourne, Australia). John is Asia Pacific IT Manager for a brand-name multi-national and he's supposed to leave all the fun technical stuff for others to do. So, his Clarion work is developed under the nom-de-keyboard Flat Chat Solutions, where "flat chat" is an Australian expression meaning doing something at top speed / high velocity.

Reader Comments

Add a comment

- » Its actually "Eggen" but that's OK. ;-) That is a common...
- » Eggad! Sorry about that Russ. Fixed. Dave
- » John, describing access to static data using the metaphor...
- » > there’s no way to force the Release() of a...

Clarion Magazine

External Business Rules with the In-Memory Driver

by Nardus Swanevelder

Published 2006-06-21

Towards the end of 2004 I wrote a series of articles on Clarion's Business rules. I discussed what you need to do to implement the standard business rule functionality in your application, and I showed how to write a template that will enable you to change your rules at runtime.

One of the problems with my approach was the fact that I used a Global queue to store the business rules, and as you are hopefully aware by now Global queues can be a problem in Clarion 6 with the new threading model.

So what to do, what to do? The one option is to encapsulate your Global Data assignments in a critical section. Various articles have been published on this subject, but the one that I found very informative and easy to implement is the use of critical sections, as discussed in Geoff Robinson's Aussie DevCon: Mambo, Critical Sections/Threading, and Ingres.

The second option you have is to use SoftVelocity's In-Memory Database Driver (IMDD), which is what I'll explore in this article. The IMDD is thread safe due to the fact that it is a file driver, and the File Manager class is thread safe. The only draw back with the IMDD driver is that you have to purchase it from SoftVelocity. It does not ship as part of Clarion Personal Edition or Clarion Enterprise Edition. The IMDD can be used for much more than replacing your Global queues with a thread safe option, but that is a discussion that does not form part of this article.

I am using the IMDD driver in most of my applications and recently I decided to upgrade my External Rules template to cater to the IMDD driver. I have left the option to use Global queues in the template for backward compatibility, but please take note that if you use the Global queues in this template, it is not thread safe.

The basics of business rules

For those of you that did not read the previous series of articles, and for those of you that can't remember them anymore, here's a quick recap.

I use three tables to store my business rules externally to my application. The first table stores the Rule System Identification Number, the Rule Description, the actual Rule Expression, the Control where you want the rule icon to be displayed and an offset for displaying the rule icon to the left of the control on the screen. Figures 1 and 2 show you sample screens of the Browse and Update screens for this table. Figure 3 gives you an example of what the rule icon looks like.

Figure 1. The Browse screen for the business rules

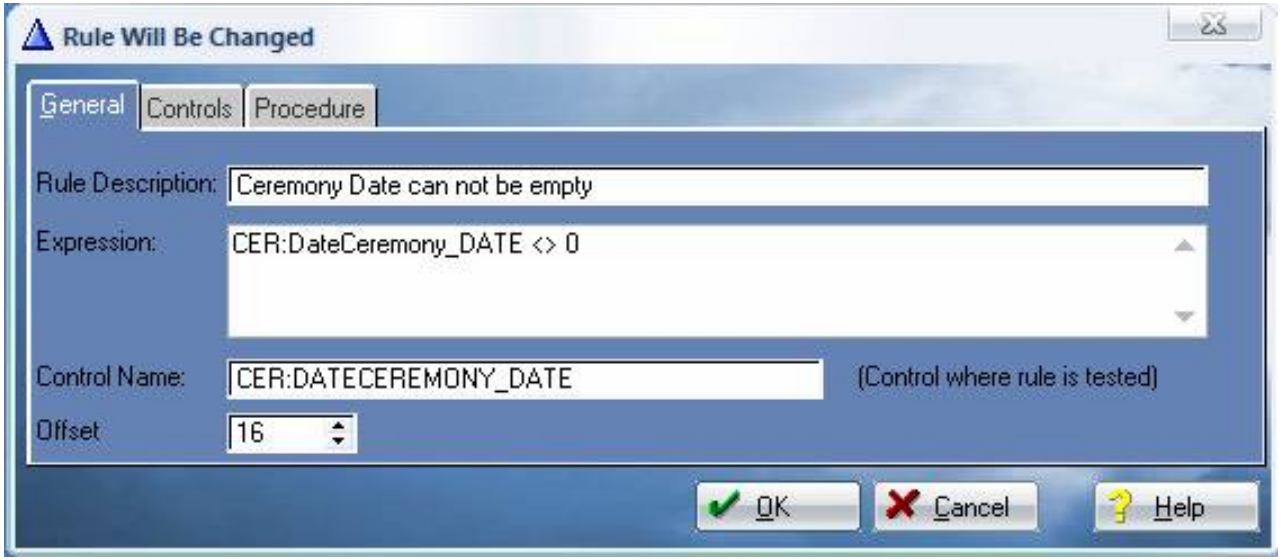


Figure 2. Example of a Rule

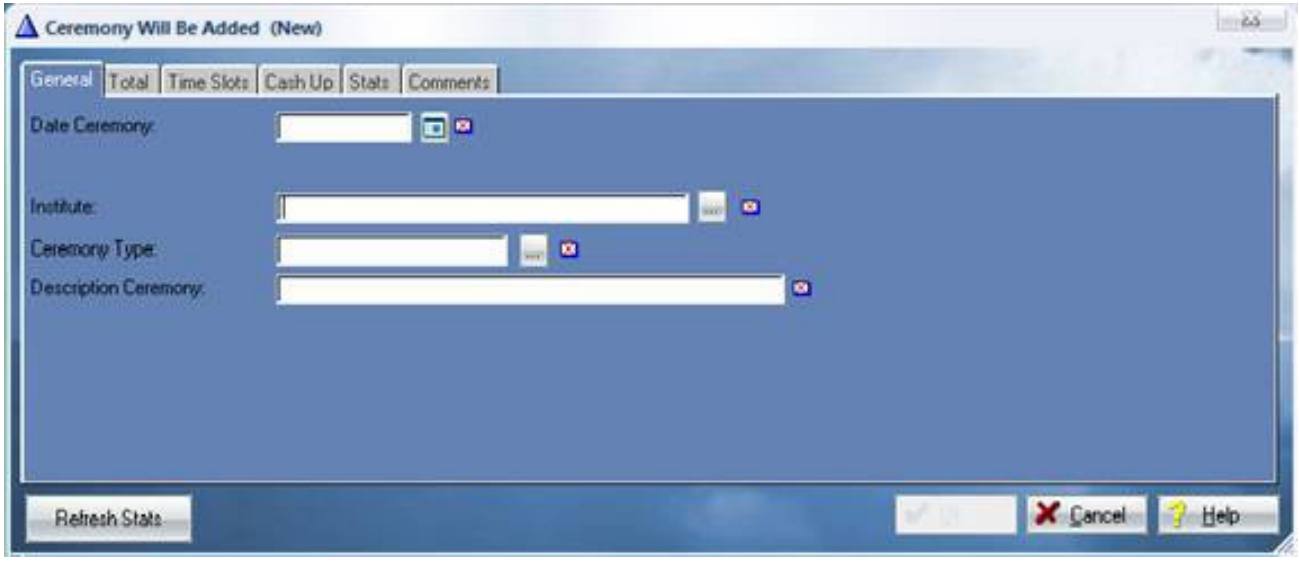


Figure 3. Update Form where there are active rules – note that the OK button is disabled

The second table stores the Action that you want to enforce if the rule is valid, and the use variable of the control to which the action will be applied. For example: if the rule is valid disable the ?OK control.

Figure 4. Example of Actions available per Control

The third table stores the names of the procedures for which the rule is excluded.

Figure 5. Example of Procedure Name

Now you know how to populate the application with your business rules, and you know what a screen will look like when the rules are enforced. The next step is to look at what is involved to add this functionality to your application.

Step 1. Add tables to your dictionary

You have to add the three tables as discussed above, and if you are going to use the IMDD, you also have to import the three IMDD tables that correspond to the queues in the non-IMDD version.

Step 2. Global Template

Add the DinamiComp's External Business Rules global template to your application's global extensions. Complete the Queue/In-Memory tab as well as the Rule Settings tab. For an explanation of the Rule Settings tab please read my previous article. In a multi-DLL application you need to add the Global extension to each of your apps where you want to test for rules on a procedure basis. The template will automatically take care of defining the external references where necessary.

Figure 6. Main Global Template

Click the Use In-Memory Driver checkbox if you are going to make use of the IMDD. Select the file IMDD Rules Table name by clicking on the button next to the entry field. I could not get the lookup button to work for a IMDD key so you have to manually enter the IMDD Rules table's primary key.

Follow this same process to complete the prompts for the Controls Per Rule and Override Procedures Per Rule tables.

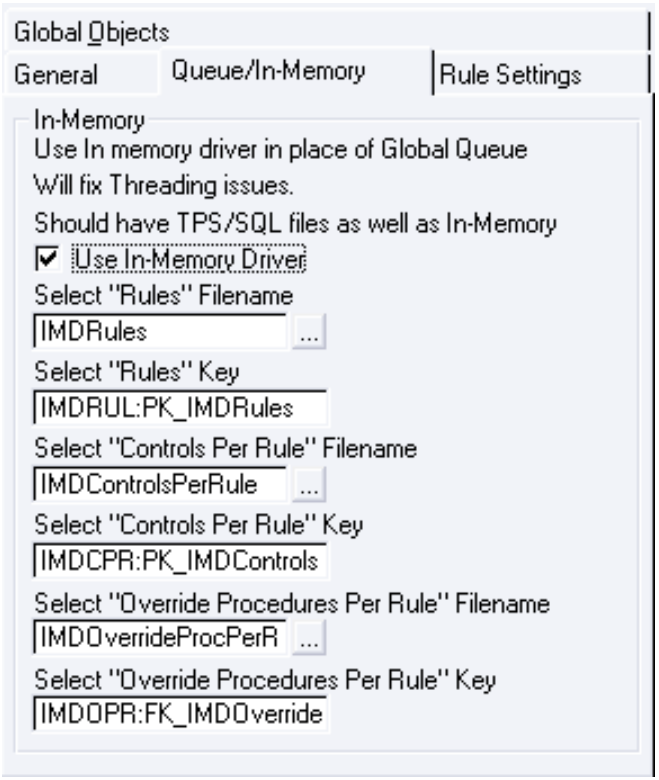


Figure 7. Population of In-Memory Data Driver Info

Step 3. Load Rules into Global Queues or IMDD table

The next step is to populate the queue or the IMDD tables with the information from the database.

To make this easier I have created a Code Extension that you call in the Main procedure of your application. Please note that this template does not make use of the InMemoryCachedTableLoad template that ships with the IMDD product. I have no experience with this table and therefore have not used it in my template.

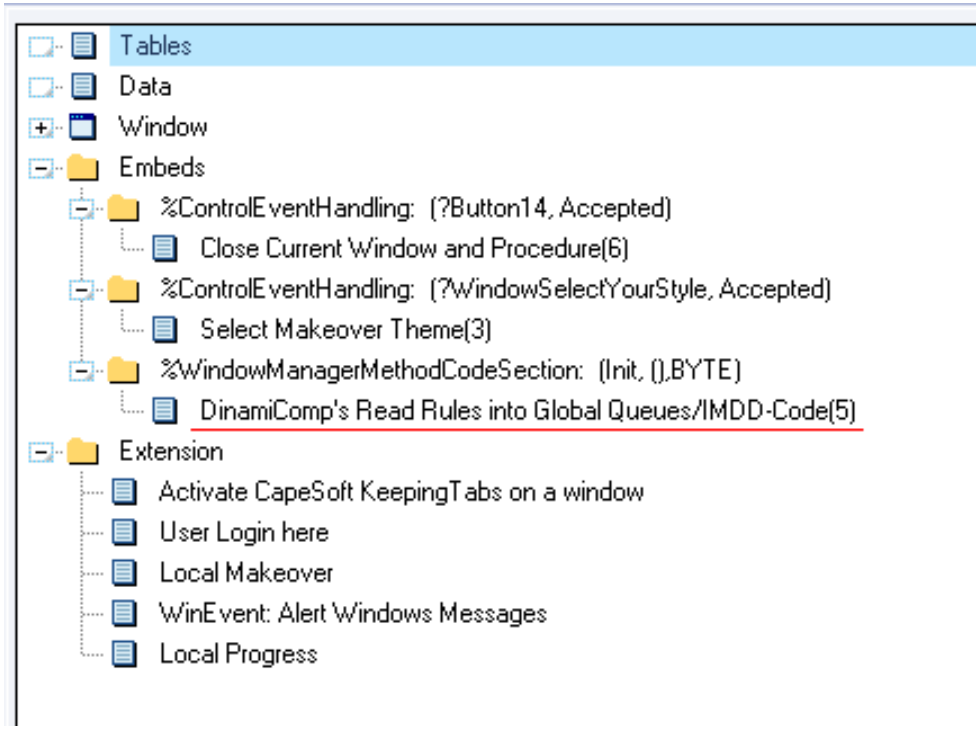


Figure 8. Read Rules from Database in to Global Queue or IMDD

Step 4. Add Procedure Extension

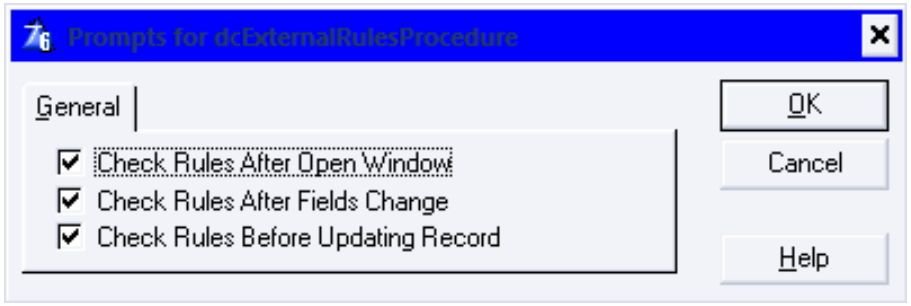


Figure 9. Procedure Template where you want rules to be checked

Step 5. Add Rules Browse and Update screens to your application

This step is only required if you believe that the client has sufficient skills/knowledge to have access to this function.

Code changes to the template

The first thing I did was to add a new #TAB to the Global Extension to be able to specify if the application is going to make use of Global queues or the In-Memory-Database-Driver (IMDD).

This is the code needed to add the new Tab (editor's note: some template lines are long - use the Printer Friendly at the top of the page to view the entire listing):

```
#TAB('Queue/In-Memory')
    #DISPLAY ('')
    #BOXED('In-Memory'),Section
       #DISPLAY ('Use In memory driver in place of Global Queue')
       #DISPLAY ('Will fix Threading issues.')
       #DISPLAY ('Should have TPS/SQL files as well as In-Memory')
       #PROMPT  ('Use In-Memory Driver',CHECK),%UseIMD,At(10,30)
       #ENABLE(%UseIMD=%TRUE)
          #DISPLAY ('Select IMDD "Rules" Filename')
          #PROMPT  ('',FILE),%IMDRules,REQ
          #DISPLAY ('Select IMDD "Rules" Key')
          #PROMPT  ('',@S255),%IMDRULEKEY,REQ
          #DISPLAY ('Select IMDD "Controls Per Rule" Filename')
          #PROMPT
('',FILE),%IMDControlesPerRule,REQ,WHENACCEPTED(%StripExclamation(%IMDControlesPerRule))
          #DISPLAY ('Select IMDD "Controls Per Rule" Key')
          #PROMPT  ('',@S255),%IMDControlesPerRuleKey,REQ
          #DISPLAY ('Select IMDD "Override Procedures Per Rule" Filename')
          #PROMPT
('',FILE),%IMDOverrideProcPerRule,REQ,WHENACCEPTED(%StripExclamation(%IMDOverrideProcPerRule))
          #DISPLAY ('Select IMDD "Override Procedures Per Rule" Key')
          #PROMPT  ('',@S255),%IMDOverrideProcPerRuleKey,REQ
       #ENDENABLE
    #ENDBOXED
#ENDTAB
```

The next step was to change the Procedure extension to be able to make use of the Global Queues or the IMDD tables.

```
  #!IMD
#Embed(%CheckIfRulesBeforeApplyingRulesB,'Check if Rules Before applying the Rules -
```

```
Begin')
  Access:%IMDRules.Open()
  Access:%IMDRules.UseFile()
  IMDRUL:SysIdRule = 0
  Set(%IMDRULEKEY,%IMDRULEKEY)
  Loop
     If Access:%IMDRules.Next() <> Level:Benign THEN BREAK.
     !Check if rule's control is on screen
     LWQ:ControlName = Upper(IMDRUL:ControlName)
     Get(LocalWindowQueue,LWQ:ControlName)
     If not error()
#Embed(%CheckProcBeforeApplyingRules,'Check Procedure Before applying the Rules')
        !If rule's control is on screen - add rule
        !Check if rule is not overridden on this procedure
        IMDOPR:SysIdRule     = IMDRUL:SysIdRule
        IMDOPR:ProcedureName = Upper('%Procedure')
        Access:%IMDOverrideProcPerRule.TryFetch(%IMDOverrideProcPerRuleKey)
        If error()
           !Procedure is not overridden - add Rule
#Embed(%CheckProcNotOverriddenApplyRules,'Check Procedure Not Overridden Apply the
Rules')

%RuleBaseName.AddRule(IMDRUL:SysIdRule,Clip(IMDRUL:RuleDescription),Clip(IMDRUL:RuleExpression)|
                                  ,LWQ:ControlEquate,IMDRUL:Offset)
           !Check rule's rule-action-control
           IMDCPR:SysIdRule = IMDRUL:SysIdRule
           Set(%IMDControlesPerRuleKey,%IMDControlesPerRuleKey)
           Loop
              If Access:%IMDControlesPerRule.Next() <> Level:Benign THEN BREAK.
              If IMDCPR:SysIdRule <> IMDRUL:SysIdRule then break.
#Embed(%CheckProcControlApplyRulesB,'Check Procedure Control Apply the Rules -
Begin')
              !If rule's rule-action-control is on window - add control to rule
              LWQ:ControlName = IMDCPR:ControlName
              Get(LocalWindowQueue,LWQ:ControlName)
              If not error()

%RuleBaseName.AddControlToRule(IMDRUL:SysIdRule,LWQ:ControlEquate,IMDCPR:RuleAction)
              End
#Embed(%CheckProcControlApplyRulesE,'Check Procedure Control Apply the Rules - End')
           End
```

```
          End
       End
    End
    Access:%IMDRules.Close()
  #Embed(%CheckIfRulesBeforeApplyingRulesE,'Check if Rules Before applying the Rules -
  End')
    #END
```

The **last step** was to populate Global Queue or IMDD tables with Rule data from database. To automate this process I created the following Code Template:

```
#!@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#!          Code Extension
#!@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#Code(dcExternalRulesCodeReadRules,'DinamiComp''s Read Rules into Global Queues/IMDD-
Code')
#!
#SHEET
    #PROMPT('Open and Close Rule
Files',Check),%LocalOpenCloseRuleFiles,AT(10),DEFAULT(1)
#ENDSHEET
 #IF(%LocalOpenCloseRuleFiles = 1)
 Access:Rules.Open()
 Access:Rules.UseFile()
 #ENDIF
 #IF(%UseIMD=%FALSE)
 Free(RulesQueue)
 RUL:SysIdRule = 0
 Set(RUL:PK_Rules,RUL:PK_Rules)
 Loop
   If Access:Rules.Next() <> Level:Benign THEN BREAK.
   RQ:SysIdRule       = RUL:SysIdRule
   RQ:RuleDescription = Clip(RUL:RuleDescription)
   RQ:RuleExpression  = Clip(RUL:RuleExpression)
   RQ:ControlName     = Clip(RUL:ControlName)
   If Sub(RQ:ControlName,1,1) = '?'
      RQ:ControlName = Sub(RQ:ControlName,2,Len(Clip(RQ:ControlName)))
   End
   RQ:Offset = Clip(RUL:Offset)
   Add(RulesQueue,RQ:ControlName)
```

```
    If error() then stop(error()).
  End
  #ELSE
  !#IMD
  Access:%IMDRules.Open()
  Access:%IMDRules.UseFile()
 #Embed(%LoadRulesInMemoryTableB,'Load Rules In Memory Table - Begin')
  RUL:SysIdRule = 0
  Set(RUL:PK_Rules,RUL:PK_Rules)
  Loop
    If Access:Rules.Next() <> Level:Benign THEN BREAK.
    IMDRUL:SysIdRule       = RUL:SysIdRule
    IMDRUL:RuleDescription = Clip(RUL:RuleDescription)
    IMDRUL:RuleExpression  = Clip(RUL:RuleExpression)
    IMDRUL:Controlname     = Clip(RUL:ControlName)
    If Sub(IMDRUL:ControlName,1,1) = '?'
       IMDRUL:ControlName = Sub(IMDRUL:ControlName,2,Len(Clip(IMDRUL:ControlName)))
    End
    IMDRUL:Offset = Clip(RUL:Offset)
 #Embed(%LoadRulesInMemoryTableBeforeAdd,'Load Rules In Memory Table - BeforeAdd')
    Access:%IMDRules.Insert()
    If error() then stop(error()).
  End
 #Embed(%LoadRulesinMemoryTableE,'Load Rules In Memory Table - End')
  Access:%IMDRules.Close()
  #ENDIF
  #IF(%LocalOpenCloseRuleFiles = 1)
  Access:Rules.Close()

  Access:ControlsPerRule.Open()
  Access:ControlsPerRule.UseFile()
  #ENDIF
  #IF(%UseIMD=%FALSE)
  Free(RulesControlQueue)
  CPR:SysIdRule            = 0
  CPR:SysIdControlsPerRule = 0
  Set(CPR:FK_ControlePerRule_Rules,CPR:FK_ControlePerRule_Rules)
  Loop
    If Access:ControlsPerRule.Next() <> Level:Benign THEN BREAK.
    RCQ:SysIdRule   = CPR:SysIdRule
    RCQ:ControlName = Clip(CPR:ControlName)
```

```
    If Sub(RCQ:ControlName,1,1) = '?'
       RCQ:ControlName = Sub(RCQ:ControlName,2,Len(Clip(RCQ:ControlName)))
    End
    RCQ:RuleAction = Clip(CPR:RuleAction)
    Add(RulesControlQueue,RCQ:SysIdRule)
    If error() then stop(error()).
  End
  #ELSE
  #!IMD
  Access:%IMDControlesPerRule.Open()
  Access:%IMDControlesPerRule.UseFile()
 #Embed(%LoadControlsPerRuleInMemoryTableB,'Load Controls Per Rule In Memory Table -
 Begin')
  CPR:SysIdRule            = 0
  CPR:SysIdControlsPerRule = 0
  Set(CPR:FK_ControlePerRule_Rules,CPR:FK_ControlePerRule_Rules)
  Loop
    If Access:ControlsPerRule.Next() <> Level:Benign THEN BREAK.
    IMDCPR:SysIdRule   = CPR:SysIdRule
    IMDCPR:ControlName = Clip(CPR:ControlName)
    If Sub(IMDCPR:ControlName,1,1) = '?'
       IMDCPR:ControlName = Sub(IMDCPR:ControlName,2,Len(Clip(IMDCPR:ControlName)))
    End
    IMDCPR:RuleAction = Clip(CPR:RuleAction)
    Access:%IMDControlesPerRule.Insert()
    If error() then stop(error()).
  End
 #Embed(%LoadControlsPerRuleInMemoryTableE,'Load Controls Per Rule In Memory Table -
 End')
  Access:%IMDControlesPerRule.Close()
  #END
  #IF(%LocalOpenCloseRuleFiles = 1)
  Access:ControlsPerRule.Close()
  Access:OverrideProcPerRule.Open()
  Access:OverrideProcPerRule.UseFile()
  #ENDIF
  #IF(%UseIMD=%FALSE)
  Free(RulesProcQueue)
  OPR:SysIdRule             = 0
  OPR:SysIDOverrideProcRule = 0
```

```
 Set(OPR:FK_OverrideProcPerRule_Rules,OPR:FK_OverrideProcPerRule_Rules)
 Loop
   If Access:OverrideProcPerRule.Next() <> Level:Benign THEN BREAK.
   RPQ:SysIdRule     = OPR:SysIdRule
   RPQ:ProcedureName = Clip(OPR:ProcedureName)
   Add(RulesProcQueue,RPQ:SysIdRule,RPQ:ProcedureName)
   If error() then stop(error()).
 End
 #ELSE
 #!IMD
 Access:%IMDOverrideProcPerRule.Open()
 Access:%IMDOverrideProcPerRule.UseFile()
#Embed(%LoadOverrideProcPerRuleInMemoryTableB,'Load Override Procedure Per Rule In
Memory Table - Begin')
 OPR:SysIdRule             = 0
 OPR:SysIDOverrideProcRule = 0
 Set(OPR:FK_OverrideProcPerRule_Rules,OPR:FK_OverrideProcPerRule_Rules)
 Loop
   If Access:OverrideProcPerRule.Next() <> Level:Benign THEN BREAK.
   IMDOPR:SysIdRule     = OPR:SysIdRule
   IMDOPR:ProcedureName = Clip(OPR:ProcedureName)
   Access:%IMDOverrideProcPerRule.Insert()
   If error() then stop(error()).
 End
#Embed(%LoadOverrideProcPerRuleInMemoryTableE,'Load Override Procedure Per Rule In
Memory Table - End')
 Access:%IMDOverrideProcPerRule.Close()
 #END
 #IF(%LocalOpenCloseRuleFiles = 1)
 Access:OverrideProcPerRule.Close()
 #ENDIF
```

Summary

In my series on Clarion Business Rules I showed how you can extend Clarion's business rules functionality to use rules defined in tables. In this way you can supply new business rules (global or local to a procedure) to your customers without having to update the application itself – instead, you simply send along an updated set of rules tables. In this article I extended the template to enable you to use the In-Memory Database Driver in stead of Global Queues, for a thread-safe solution.

Download the source

Nardus Swanevelder was born and raised in South Africa. He was a networking engineer for seven years before he moved over to the commercial side of the business. Nardus has developed a Sale Cycle Management system for the Information and Communication Technology industry. He has been programming in Clarion since 1989, and holds B.Com and MBA degrees. In his spare time Nardus lectures Financial Management to B. Com Hons students at North-West University.

Reader Comments

Add a comment

Clarion Magazine

Using RMChart with Clarion

by Al Randall

Published 2006-06-15

One of the nice-to-have features in any modern program is the ability to display charts. After researching several alternatives I happened upon RMChart. This article describes how you can incorporate charting using their OCX in a Clarion program.

RMChart is a "wrapper" around the Microsoft GDIPlus.DLL. GDIPlus is an installed part of WinXP located in your Windows\System32 folder. If you do not have WinXP, it is available as a separate download from the Microsoft website for other operating systems. This article will assume you either already have the GDIPlus.DLL. As I'll show later, you can still run the rest of your application if the user's system doesn't have GDIPlus.

The RMChart download installs RMCDesigner, RMChart.DLL and RMChart.OCX. RMCDesigner is a user-friendly program to allow you or your users to design a chart; you can incorporate test data and save the result as an RMC file.

While you can build the chart at run time from the ground up, I elected to design a basic chart using RMCDesigner and fill in the data and labels at run time. This approach seemed to make most sense to me since my users can then "tweak" the chart with colors, legends, etc, etc. This keeps them involved in the chart design and they love that feeling!

RMChart supports two methods of incorporating charting, DLL and OCX. There are advantages and disadvantages to both approaches but for my purposes, I settled on using an OCX since an OCX gave me more control over placement and size of the resulting chart and allowed me to place additional data fields on the window outside of the chart area.

Creating charts

Download and install the complete RMChart installation package. This is a free download from the

author's web site.

Use RMCDesigner to design your basic chart. The package comes with a number of pre-designed charts which you can use to get started. I suggest starting with a simple line or bar chart. Save the chart in your application folder as "xxxxxx.RMC". You'll load it later when you begin building the chart with Clarion. Think of this as your "chart template." The example chart available in the source download uses the three basic RMC templates (Simpleline.RMC, Simplebar.RMC and Simpledonut.RMC) installed as part of the RMChart installation.

On a standard Clarion Window, place an OCX and make certain it is set to 32 bit and OCX. I left the object type blank as I create it at run time.

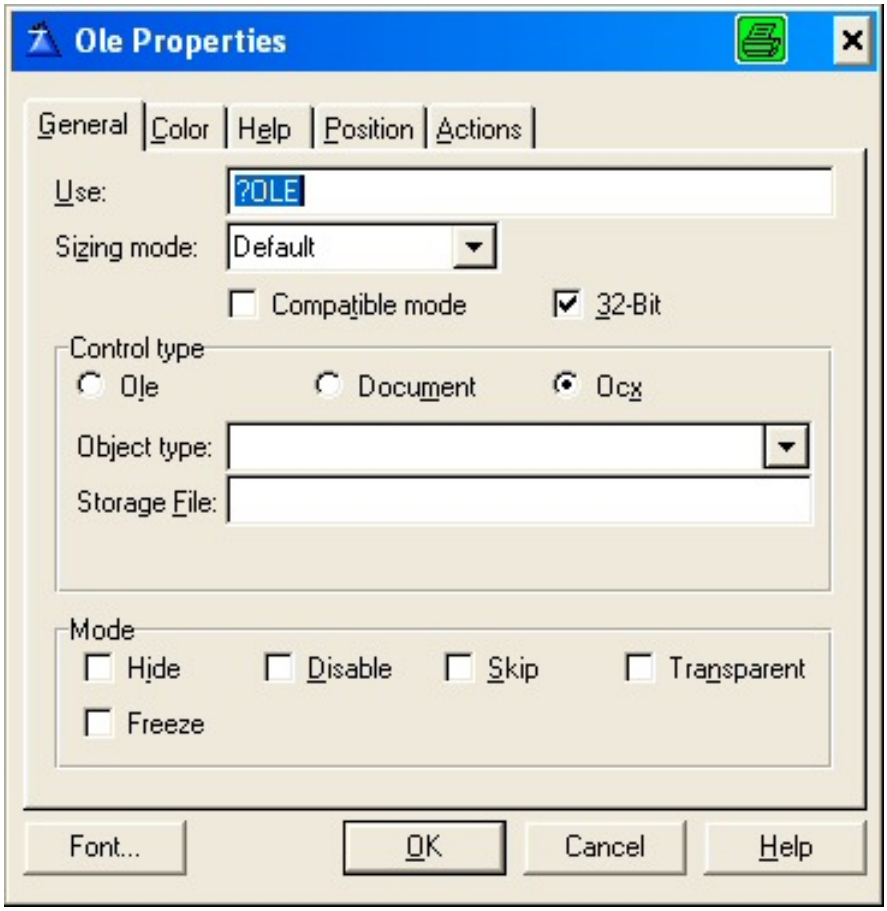


Figure 1. Add the OCX to your window

In your data section, define the variables you'll use to chart. In this example I'm building a line chart with five different lines.

```
! Declare values for Charting
LP:CxPos GROUP,PRE()
LP:CxX  LONG    ! X position in Pixels
LP:CxY  LONG    ! Y position in Pixels
LP:CxW  LONG    ! Width in Pixels
LP:CxH  LONG    ! Height in Pixels
```

```
          END
LP:CxDataPoints  SHORT  ! Number of data points
LP:CxLblString   CSTRING(501)   ! String for Labels
LP:CxDataStrings  GROUP,PRE()
LP:CxDataString1  CSTRING(501)  ! String to hold data values 1
LP:CxDataString2  CSTRING(501)  ! String to hold data values 2
LP:CxDataString3  CSTRING(501)  ! String to hold data values 3
LP:CxDataString4  CSTRING(501)  ! String to hold data values 4
LP:CxDataString5  CSTRING(501)  ! String to hold data values 5
     END
```

Define a memory queue to hold a sort field, a label field and one or more data fields. In this example, I'm building a line graph with five different lines. While there are a number of ways to accomplish this, I elected to place my data into a memory queue called `Cxq:Queue`. I chose this approach so that I could eventually write a Clarion template and simply populate a queue and pass it to a standardized routine. Using this approach I defined my queue with a sort field, a label field and one or more data fields.

```
!Queue to contain chart X-Axis Labels and Data
CxQ:Queue      QUEUE,PRE()
CxQ:Record       GROUP
CxQ:Sort         STRING(100)  ! Sort value
CxQ:LabelValue   STRING(100)  ! Label Value
CxQ:DataValue1   LONG         ! Data Value 1
CxQ:DataValue2   LONG         ! Data Value 2
CxQ:DataValue3   LONG         ! Data Value 3
CxQ:DataValue4   LONG         ! Data Value 4
CxQ:DataValue5   LONG         ! Data Value 5
               END
             END
```

At an embed point after opening your window, you'll need to create the OCX. In my case, I incorporated some paranoid logic to verify that the user had the necessary DLL and OCX installed. Initially I also tested for the existence of the GDIPlus.DLL, but quickly discovered that it could exist in a number of locations so I decided against that test. If you do want to go this route, note that the standard RMChart install process installs RMChart.DLL and RMChart.OCX in the Windows/System32 folder and registers the OCX for you.

In the following code I get the position of the OLE control (note setting PROP:Pixels true first). I need to know the control's position since I will resize the chart at run time to fill the control, and then reposition the chart at the point I draw it. I also placed my chart on a hidden tab (`?GraphTab`) and only show the tab if the OCX is present; that way if the necessary components aren't installed I don't have users asking "Why can't I see the chart?"

```
! Does the necessary Charting OCX/DLL exist?
CLEAR(LP:CxPos)
```

```
IF EXISTS('C:\WINDOWS\SYSTEM32\RMChart.OCX') AND |
    EXISTS('C:\WINDOWS\SYSTEM32\RMChart.DLL')
  ?OLE{PROP:Create} = 'RMChart.RMChartX'
  IF ?OLE{PROP:OLE}
    0{PROP:Pixels} = TRUE
    GETPOSITION(?OLE,LP:CxX,LP:CxY,LP:CxW,LP:CxH)
    0{PROP:Pixels} = FALSE
  ELSE
   ! Hide the control
   ?GraphTab{PROP:Hide} = TRUE
  END
ELSE
 ! Hide the control
 ?GraphTab{PROP:Hide} = TRUE
END
```

To build and display the chart, I load the chart template created earlier.

I placed this code after opening the window and data files. You can also title the chart and change the chart legends at run time as I've demonstrated below. Note that all data strings passed to the charting OCX are asterisk delimited. In the following there are five values passed to the `LegendString` property.

```
! Load the RMC file; title the graph
?OLE{'RMCFile'} = 'xxxxx.rmc'
! Note "Titel" and NOT "Title"
?OLE{'Region(1).Caption.Titel'} = 'Title of the chart'
?OLE{'Region(1).Legend.LegendString'} = |
    'OnTime Pcnt*87% Target*Supplier Rating*Fill Rate*NCMR'
```

Resize the chart to fit the OCX area. Note that this property expects sizes in pixels which is why the code calls `GETPOSITION` *after* setting `PROP:Pixels` to `True`.

```
! Resize graph to fill control area
?OLE{'RMCWidth'}  = LP:CxW
?OLE{'RMCHeight'} = LP:CxH
```

Load the data values and save them in the chart queue.

```
FREE(Cxq:Queue)
! Build your data record and save in the queue
LOOP <until data queue filled>
   CxQ:Sort       = <Sort field>                    ! String
   CxQ:LabelValue = <Label Field>              ! String
   CxQ:DataValue1 = <Numeric Value 1>    ! Data values
   CxQ:DataValue2 = <Numeric Value 2>    ! Data values
```

```
    CxQ:DataValue3 = <Numeric Value 3>    ! Data values
    CxQ:DataValue4 = <Numeric Value 4>    ! Data values
    CxQ:DataValue5 = <Numeric Value 5>    ! Data values
    ADD(CxQ:Queue,+CxQ:Sort)
END
```

Once you've loaded your data into the queue, sort the queue and build the label and data strings to pass to the chart. Remember you're building an *asterisk* delimited string for both chart labels and data. Note the use of the *sort* variable and *label* variable. In some cases you may need to sort the queue in one sequence but wish it labeled in another. For example you may wish to sort the data queue in year/month sequence (0101; 0102; 0103; 0104) but want the labels in another (Jan 01; Feb 01; Mar 01; Apr 01).

```
! Sort data queue into ascending sequence
SORT(CxQ:Queue,+CxQ:Sort)
! Build data and label strings
CLEAR(LP:CxLblString)   ! Axis labels
CLEAR(LP:CxDataStrings) ! Data strings
CLEAR(LP:CxDataPoints)  ! Number of data points
! Read from lowest to highest
LOOP LP:CxDataPoints = 1 TO RECORDS(CxQ:Queue)
  GET(CxQ:Queue,LP:CxDataPoints)
  IF NOT ERRORCODE()
  ! Is this first time?
    IF LP:CxDataPoints=1
      ! First time
      ! Use either Label value or Sort Value
      IF CxQ:LabelValue
        LP:CxLblString = CLIP(CxQ:LabelValue)
      ELSE
        LP:CxLblString = CLIP(CxQ:Sort)
      END
      ! Load the data into the strings; make the strings "compact"
      LP:CxDataString1 = CLIP(LEFT(FORMAT(CxQ:DataValue1,@N_8),8))
      LP:CxDataString2 = CLIP(LEFT(FORMAT(CxQ:DataValue2,@N_8),8))
      LP:CxDataString3 = CLIP(LEFT(FORMAT(CxQ:DataValue3,@N_8),8))
      LP:CxDataString4 = CLIP(LEFT(FORMAT(CxQ:DataValue4,@N_8),8))
      LP:CxDataString5 = CLIP(LEFT(FORMAT(CxQ:DataValue5,@N_8),8))
    ELSE
      ! Not first time
      IF CxQ:LabelValue         ! Use either Label value or Sort Value
        LP:CxLblString = LP:CxLblString & '*' & CLIP(CxQ:LabelValue)
      ELSE
        LP:CxLblString = LP:CxLblString & '*' & CLIP(CxQ:Sort)
      END
      ! Load the data into the strings; make the strings "compact"
      LP:CxDataString1 = LP:CxDataString1 & '*' |
```

```
          & CLIP(LEFT(FORMAT(CxQ:DataValue1,@N_8),8))
        LP:CxDataString2 = LP:CxDataString2 & '*' |
          & CLIP(LEFT(FORMAT(CxQ:DataValue2,@N_8),8))
        LP:CxDataString3 = LP:CxDataString3 & '*' |
          & CLIP(LEFT(FORMAT(CxQ:DataValue3,@N_8),8))
        LP:CxDataString4 = LP:CxDataString4 & '*' |
          & CLIP(LEFT(FORMAT(CxQ:DataValue4,@N_8),8))
        LP:CxDataString5 = LP:CxDataString5 & '*' |
          & CLIP(LEFT(FORMAT(CxQ:DataValue5,@N_8),8))
      END
    ELSE
      ! Error, simply pass blanks and zeros in strings
      LP:CxLblString = LP:CxLblString & '* '
      LP:CxDataString1 = LP:CxDataString1 & '*0'
      LP:CxDataString2 = LP:CxDataString2 & '*0'
      LP:CxDataString3 = LP:CxDataString3 & '*0'
      LP:CxDataString4 = LP:CxDataString4 & '*0'
      LP:CxDataString5 = LP:CxDataString5 & '*0'
    END
  END
```

Once you've built your label and data strings, you'll need to set the appropriate property; reposition the OLE for "insurance" and tell the OCX to draw the chart.! Pass the data values to the graphing routine and draw the graph:

```
  ?OLE{'Region(1).LabelAxis.TickCount'}   |
    = LP:CxDataPoints      ! Number of data points
  ?OLE{'Region(1).LabelAxis.LabelString'} |
    = LP:CxLblString       ! Set the labels
  ?OLE{'Region(1).LineSeries(1).DataString'} |
    = LP:CxDataString1  ! Line Chart
  ?OLE{'Region(1).LineSeries(2).DataString'} |
    = LP:CxDataString2  ! Line Chart
  ?OLE{'Region(1).LineSeries(3).DataString'} |
    = LP:CxDataString3  ! Line Chart
  ?OLE{'Region(1).LineSeries(4).DataString'} |
    = LP:CxDataString4  ! Line Chart
  ?OLE{'Region(1).LineSeries(5).DataString'}|
     = LP:CxDataString5  ! Line Chart
  ! Reposition the control - just in case
  0{PROP:Pixels} = TRUE
  SETPOSITION(?OLE,LP:CxX,LP:CxY,LP:CxW,LP:CxH)
  0{PROP:Pixels} = FALSE
  ?OLE{'Draw'}
```

Finally, deactivate the OCX and free the charting queue when exiting the program

```
IF ?OLE{PROP:Create}
  ?OLE{PROP:Deactivate} = TRUE
  FREE(CxQ:Queue)
END
```

Chart styles

Depending on the type of chart you're using, you'll need to address different properties when setting these values. The example above is a line chart. For bar charts, you'll pass values as "BarSeries(1)" and for pie charts as "GridlessSeries". Note that line and bar graphs can have multiple indicies (e.g. (1), (2), (4)) while pie charts do not. Also note that labels must be addressed differently with Pie vs Line/Bar charts.

```
! Set the labels for Line/Bar chart
?OLE{'Region(1).LabelAxis.LabelString'} = LP:CxLblString
! Set the Pie chart labels
?OLE{'Region(1).Legend.LegendString'} = LP:CxLblString
! Set the data for Line Chart
?OLE{'Region(1).LineSeries(1).DataString'} = LP:CxDataString1
! Set the data for Bar Chart
?OLE{'Region(1).BarSeries(1).DataString'} = LP:CxDataString1
! Set the data for Pie Chart
?OLE{'Region(1).GridlessSeries.DataString'} = LP:CxDataString1
```

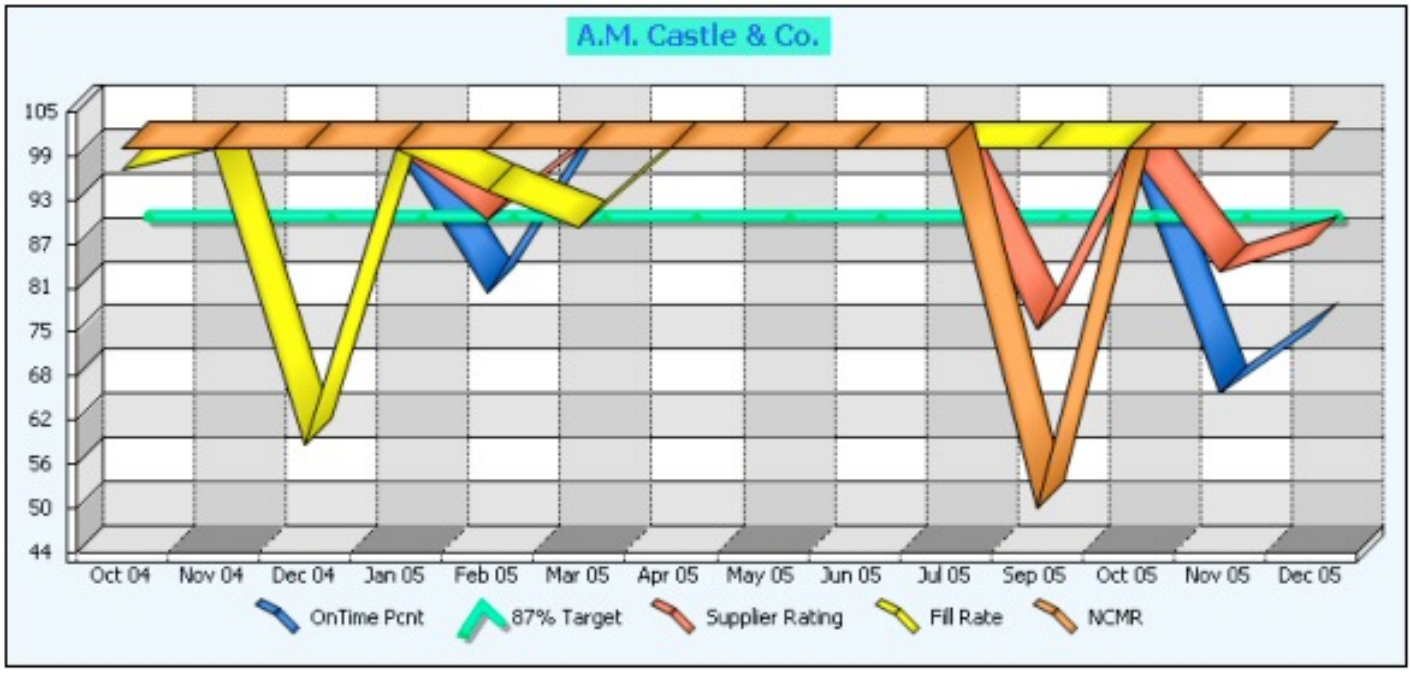


Figure 2. The completed chart

Take it from here

I've outlined the use of the RMChart OCX approach to adding charts to your Clarion program. The software is quite flexible and can create multiple "regions" (charts) within one control. As noted you can have multiple series (lines or bars) within a single region. I urge you to spend some time with the RMCDesigner to see what it can accomplish. To understand the syntax necessary to create the chart, I suggest building the chart the way you would like to see it and then using the "Copy source to clipboard" facility using the Visual Basic 6 with Extended Source style. Study the source a bit and compare it with the syntax as outlined in this article and all should become clear.

Download the source

Al Randall moved from a mainframe systems programming environment to management, then to midrange business applications, and finally landed in the PC world as an independent developer back when the IBM XT was state-of-the-art. In the PC environment Al developed in dBase, FoxPro and Magic before discovering Clarion Professional Developer 2.1. Since then he has worked with Clarion for Windows 1.5 and 2, and most recently Clarion 5. Al has found very little that he can't accomplish with that platform, including machine tool control and business applications for a variety of industries.

Reader Comments

Add a comment

Clarion Magazine

Clarion 7 and Clarion.NET: Video No 1

by Dave Harms

Published 2006-06-12

SoftVelocity has released a screencast of some features of Clarion 7. You can view a low-medium bandwidth version or a high bandwidth version. A PDF of the PowerPoint slides is also available. It will most likely take a few minutes after you click on the link for the screencast to load.

This screencast, by Bob Zaunere, is mainly about the full XP Themes support in C7, although Bob does make a few points about other C7 features, and Clarion.NET.

Bob begins by talking about the new IDE, and emphasizes that there is only one IDE for both C7 and Clarion.NET. This IDE is, as has been previously noted, a marriage between the Clarion IDE and the SharpDevelop IDE. SharpDevelop is a popular alternative to Visual Studio.

SV has a commercial code license for the SharpDevelop IDE. There's been some confusion over this in the past, since #develop (as it's also known) is available as open source. What's sometimes missed is that the creators of #develop are free to release code under other licenses as well, which is what they've done with SoftVelocity (and perhaps other vendors). Also note that SV reportedly gets updated versions of #develop; this means that features added by the #develop team should make their way into the SV IDE as well, where practical.

Back to the presentation – Bob makes the point that migrating from C6 to C7 should be completely painless. The C6 and C7 code bases are being kept in sync, so any C6 fixes are immediately applied to C7. Of course, there are C7 features (full support for XP Themes)

which aren't in C6, and it's apparently from the latest release of the C6 templates that this code is simply omitted from C6 compiles by the use of compiler switches.

As well, the new IDE works with all version of Clarion back to Clarion 4. This is something that's been talked about before, and I imagine it's not that trivial a task. Even though we haven't seen the new IDE yet, the fact that this feature is still there, and hasn't been quietly dropped, suggests that those technical challenges have been largely overcome. That's my assessment only.

The point of being able to use older version of Clarion (the old templates, the old compiler, and presumably any related third party stuff) with the new IDE is that you gain the advantages of the new developer UI, including better editors, flatter interface, etc.

Project system

With C7 you load an app or project, and at that time you can decide which version of Clarion to use (see Figure 1).

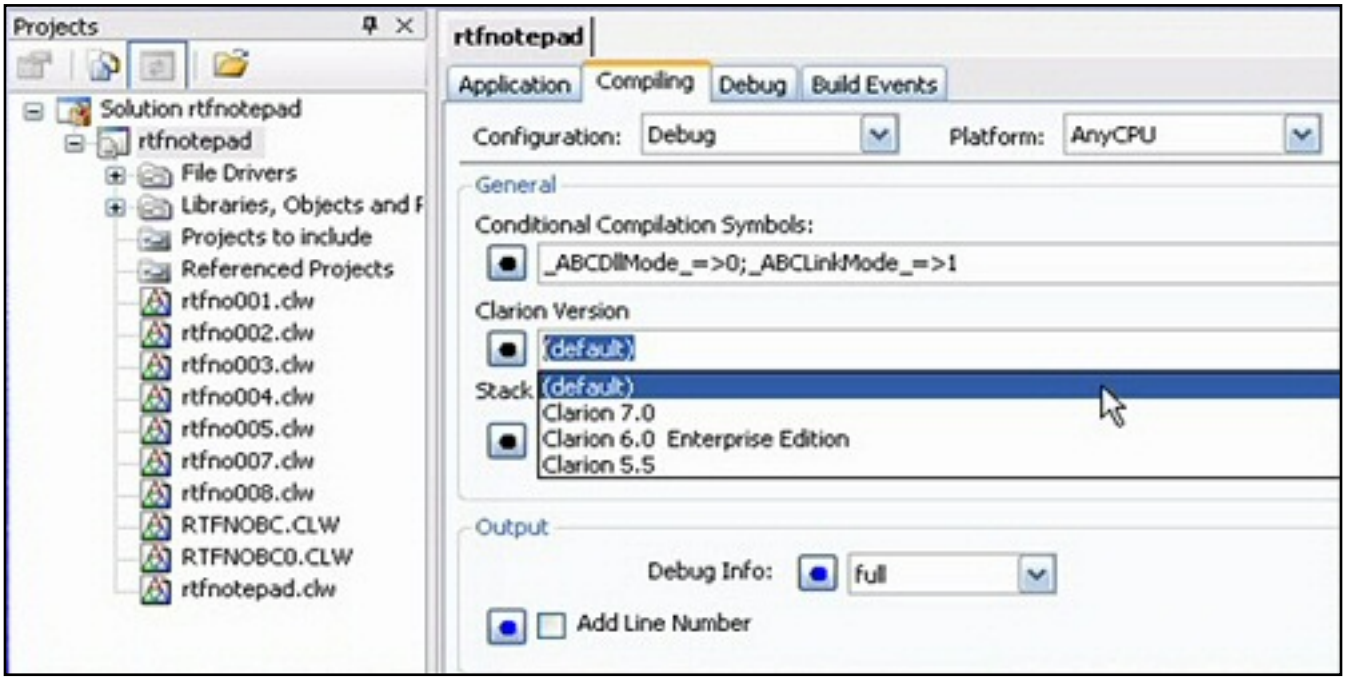


Figure 1. Loading an app in C7

For Clarion 7 (and Clarion.NET) the project system has been changed so that it uses (and apparently extends) Microsoft's new .NET-based, XML-based MSBuild system. Microsoft has a web page and a blog for MSBuild. It looks like the C7/Clarion.NET build

system will add some nifty features (Bob mentions solution files containing multiple apps, where building the solution triggers automatic code generation for the APP files) and presumably there will also be lots of room for enhancements by Clarion third party vendors. You can add your own specialized build tasks, and the MSBuild system can also be used from the command line..

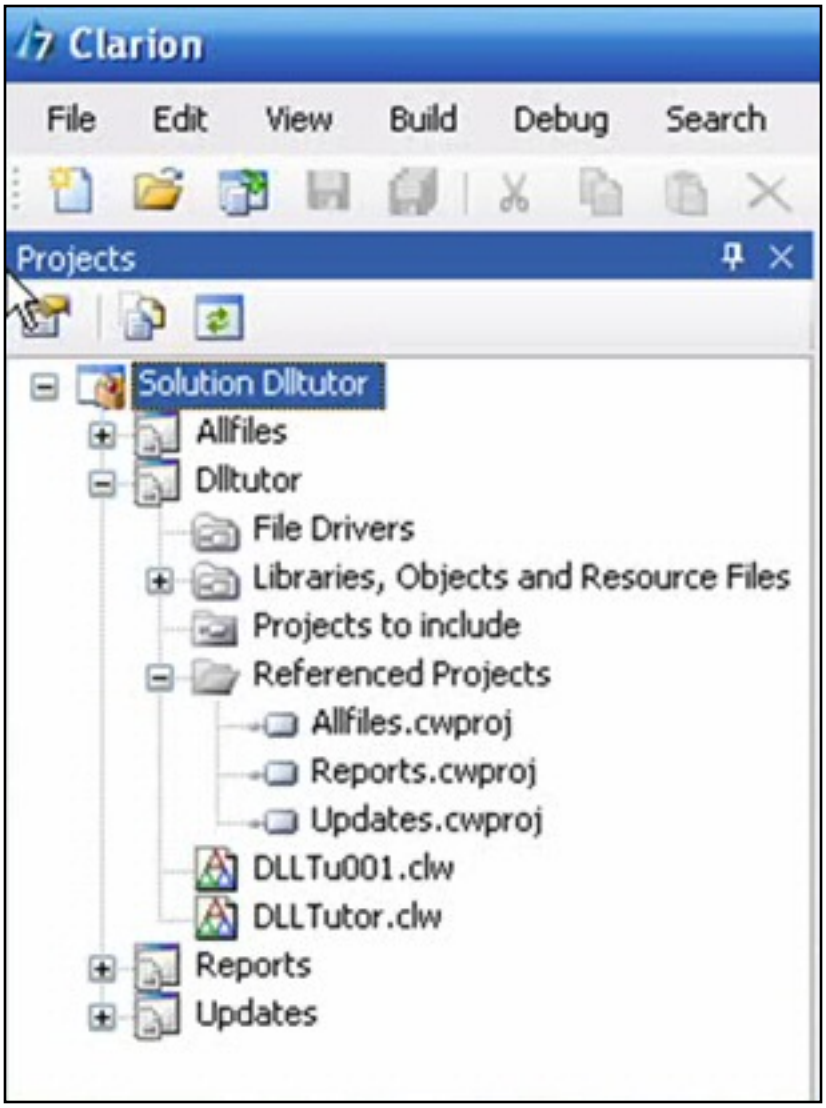


Figure 2. A C7 solution

You can build both debug and release configurations with a single solution.

XP Themes

Bob makes the point that with full threading, Clarion developers are already creating some of the most sophisticated business applications on the planet. Version 7 delivers on improvements your end users can see.

In C7 every window and control has support for XP Themes, even controls (such as menus) which don't, under Windows, normally have full support. More on that below.

Tabbed MDI

C7 adds support for tabbed MDI windows. This is something you (and your users) have probably seen in FireFox, or in IE7.

Figure 3. Four styles of tabbed MDI windows

C7 offers four different styles of MDI tabs, as you can see in Figure 3. Note that when tabs are colored, the color doesn't extend into the tab body; instead, the tab body takes on the XP Theme color for that control. Figure 4 shows the old style tabs on the left, the new style on the right.

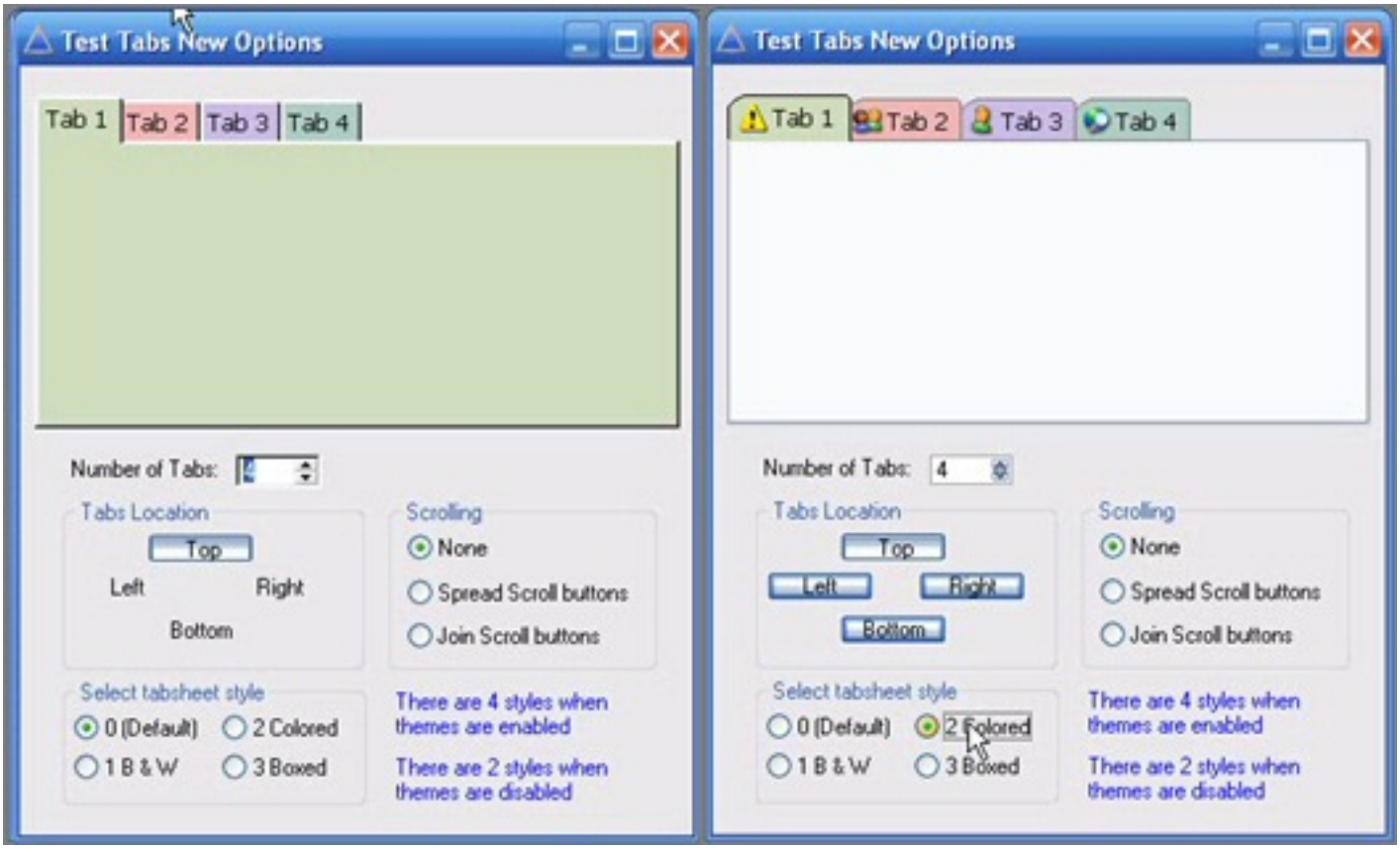


Figure 4. C6 (left) and C7 (right) colored tabs

Bob's example of tabbed MDI shows different sort orders, but to me that's not so much different from using tabs on a browse. I think using tabbed MDI to show different browses is a better example. One thing to keep in mind about tabbed MDI is that you're full screen on each tab, so presumably, in the MDI way of things, your form is going to be full screen too. I guess we'll have to see how well this works with forms and browses.

Note that MDI tabs are scrollable if there are more tabs than will fit. You can also close the selected window, and since these are regular windows you can select a particular tab via the regular Windows list. You can also switch the tab toolbar on and off at runtime.

XP Themes on tabs are preserved when the tabs are on the sides or bottom of the sheet as well.

C7 adds XP Theme support to menus as well (Figure 5).

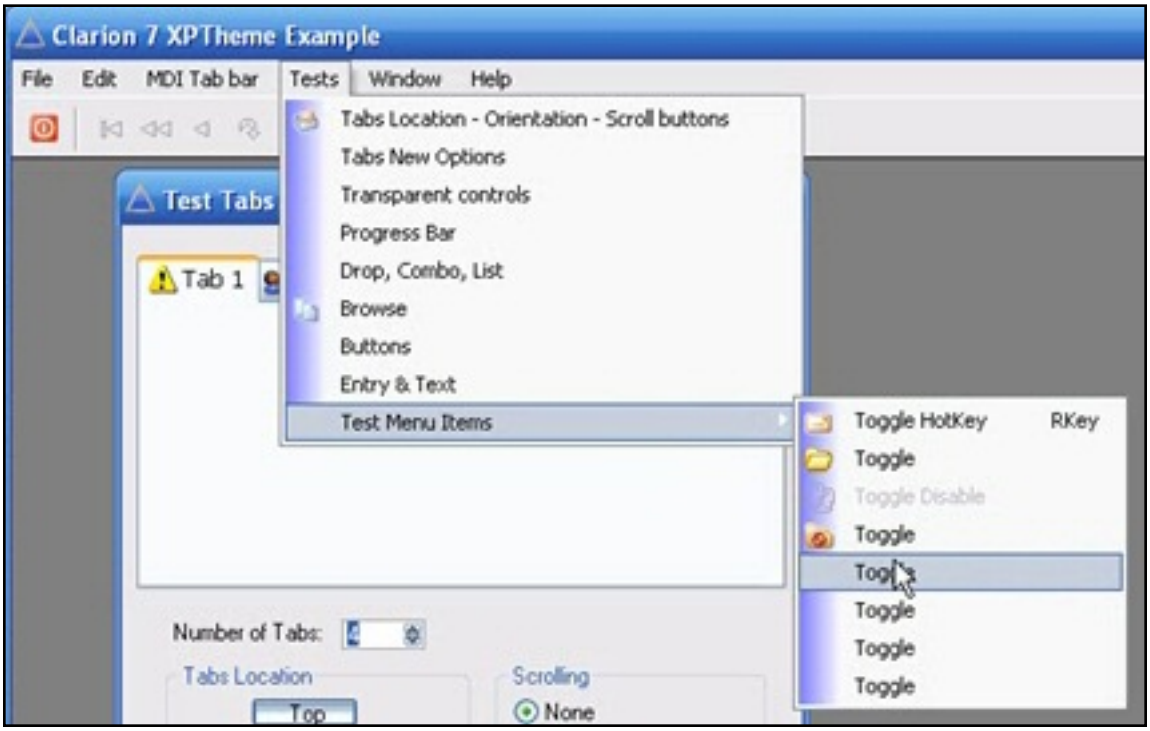


Figure 5. XP Themes in menus

The partial XP Theme support for browses in C6 (see Figure 6) has been updated to full support (Figure 7).

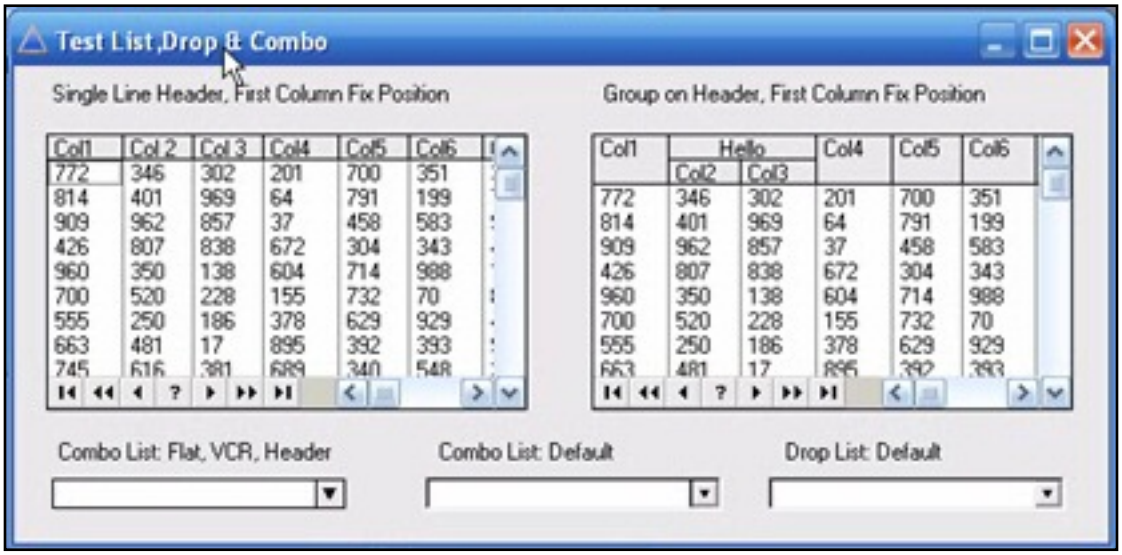


Figure 6. C6 browse theme support

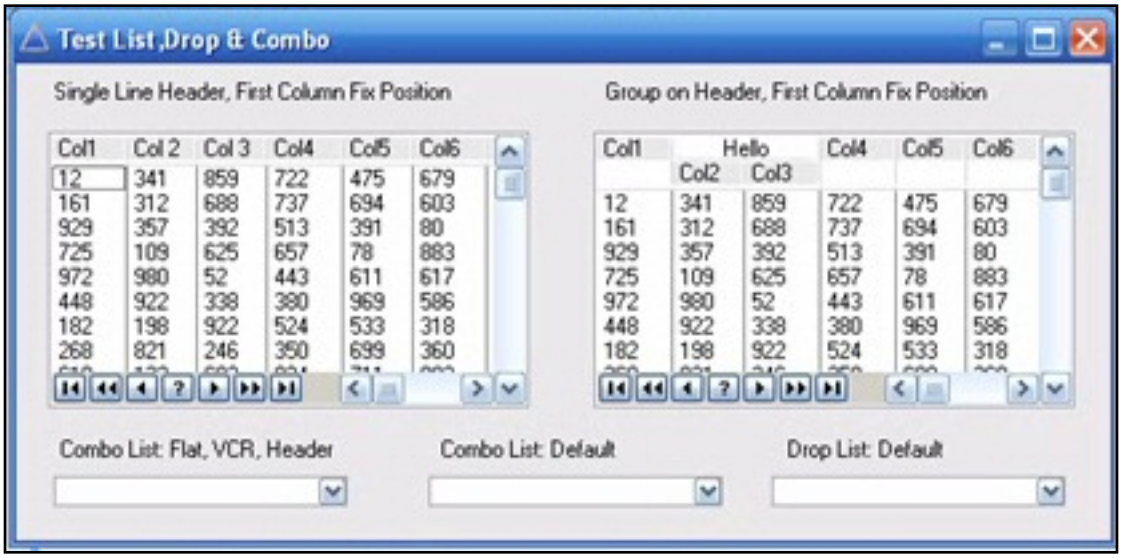


Figure 7. C7 browse theme support

Entry and text controls also get the full treatment, although the differences here aren't as dramatic (Figure 8).

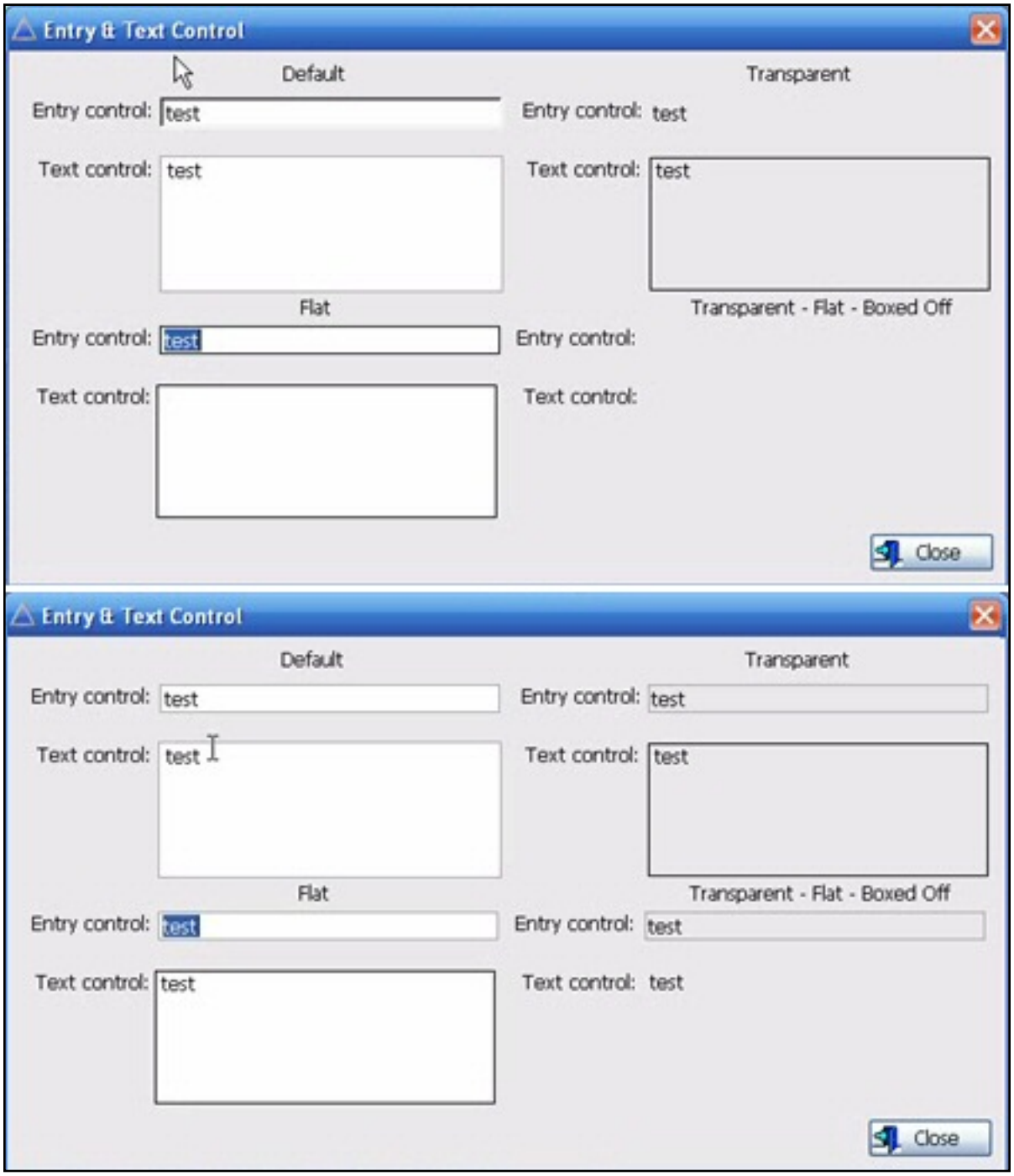


Figure 8. Entry and text theme support, C6 (top) and C7 (bottom)

Menus can include icon animation for the currently selected item, something that isn't that easy to see in the demo.

Not shown in the demo, but mentioned, is a new XP style toolbox with gradients and grips. There is also some new code for creating disabled or grey-scaled icons, and C7 will have support for PNG images and the RichEdit 4.1 control, including clickable URLs.

IDE changes

There are numerous changes to the dictionary editor, including graphical views of tables and relationships (Enterprise Edition only), grouping features, a global view for global edits, cascading field changes, the ability to edit multiple tables at once, personalized layouts of views, fields and keys, new edit-in-place functionality, and a variety of reporting options. No screen shots yet.

The AppGen has been completely refactored for the fastest possible code generation speed. It has better procedure and class views, and new searching capabilities. And you can have multiple app files open at once.

Source editor

Productivity features in the new source editor include code folding, a class browser, procedure and method navigators (that will take you to the corresponding source files), search/replace across projects and solutions, and access to the dictionary from hand code.

Templates

New templates add support for toolboxes and the tabbed MDI feature, and there is a new generation template to assist in writing classes.

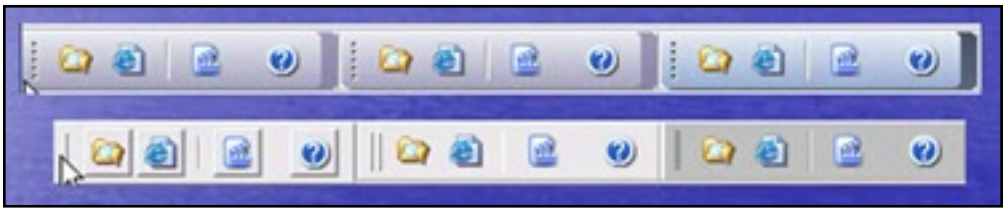


Figure 9. The new toolbox styles

The formatters

The window and report formatters are new, and something that's a bit of a surprise to me is that the same formatters are used for Win32 and for Clarion.NET (Figures 10 and 11). As you'd expect, the property grid can be moved around and resized.

Actually this approach makes a lot of sense, since it's probably a lot easier to create a set of .NET controls that can be mapped at code generation time to Win32 controls, than it is to maintain two formatters. That's just my assumption; I don't know just how SV has implemented the Win32 aspect of the formatters.

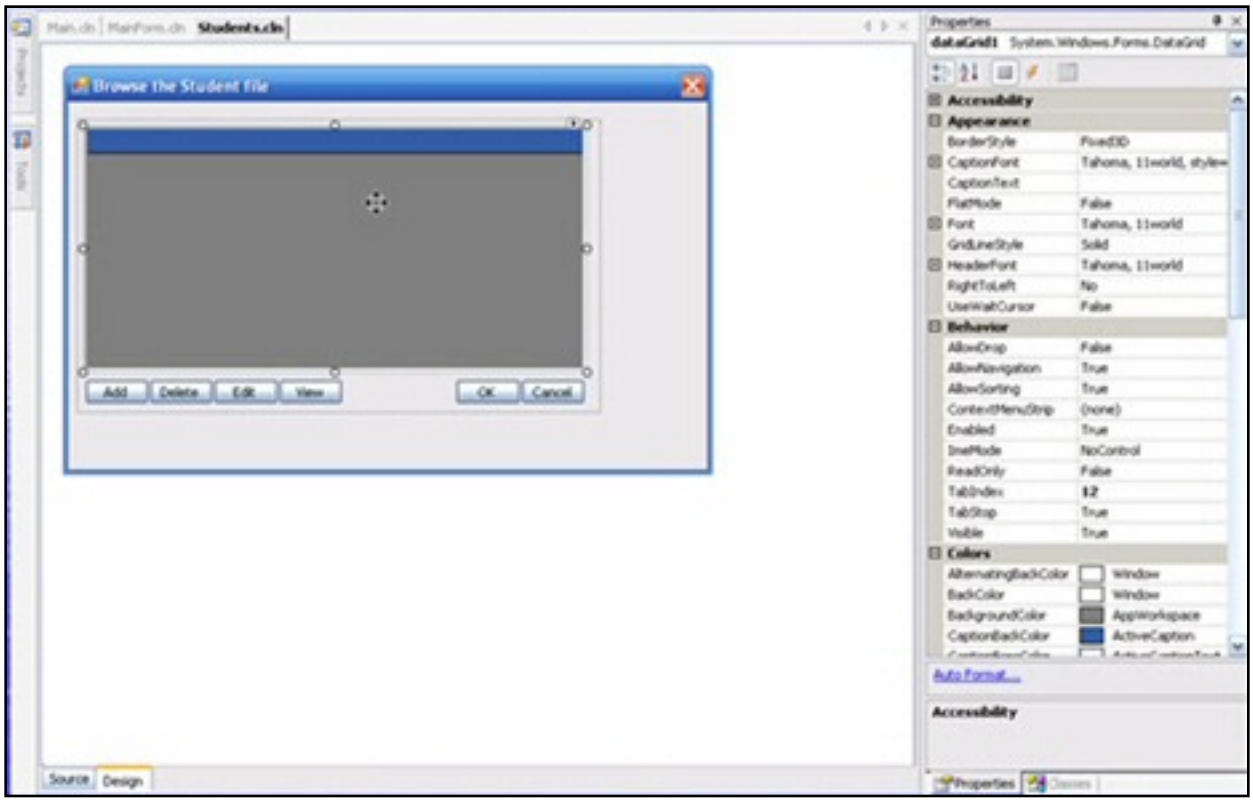


Figure 10. The new Window formatter

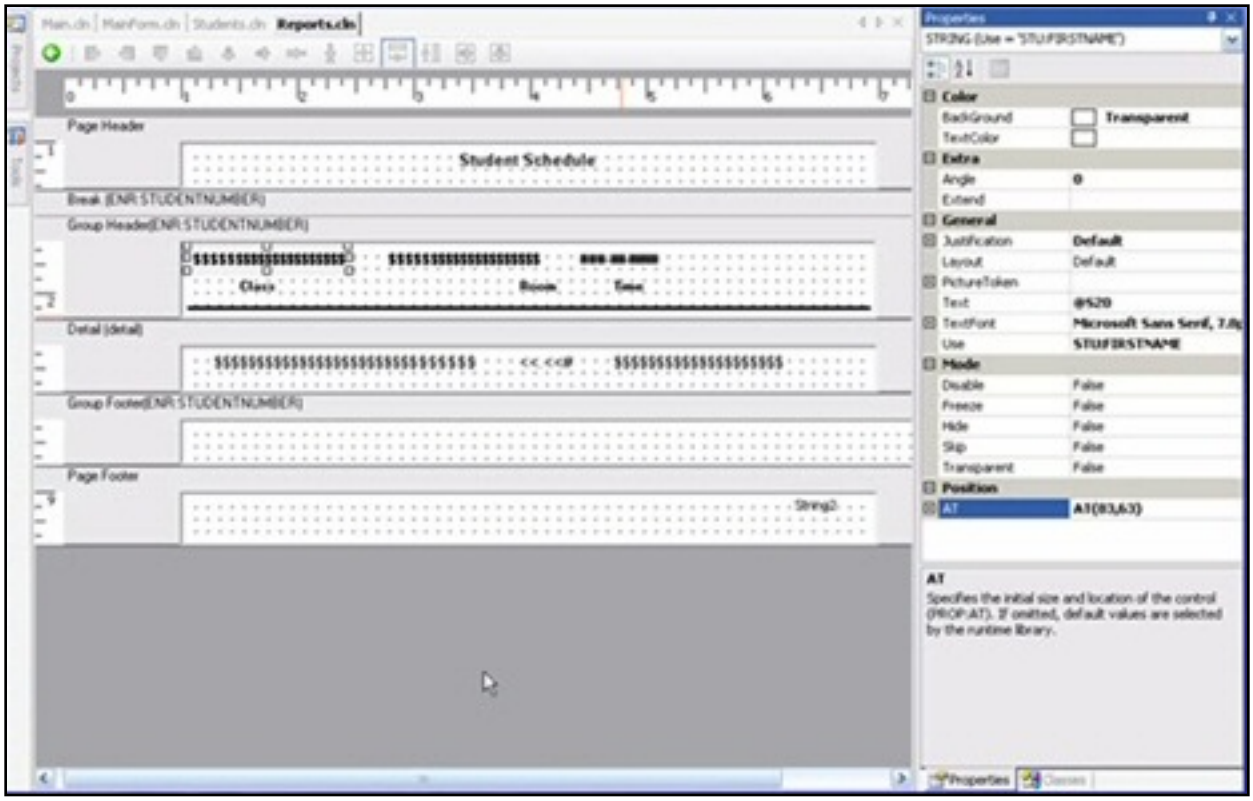


Figure 11. The new Report formatter

Conclusions

I'm glad to get a more detailed look at what's coming in C7 and Clarion.NET. I'd still like to see the new IDE itself; both the new dictionary editor and the flattening of the template interface in AppGen should be dramatic departures from what we're used to.

On the other hand, one of the lasting complaints about Clarion, over the many years I've been using the product, is that while you can get a lot done fast, the results aren't very pretty.While there are third party products that can do the job, this is something that should be available out of the box. Get ready to cross the "but it ain't pretty" objection off the list.

On its own, full XP Theme support is a much-needed step forward; combine that with the promised productivity enhancements in the project system, dictionary editor and AppGen, and the decision to upgrade to C7 should be a no-brainer.

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 produces the Planet Clarion podcast, which he co-hosts with Andrew Guidroz II.

Reader Comments

Add a comment

- » Hi Dave, I think you may have used they C6 screen grab...
- » Hi Dave, I think you may have used they C6 screen grab...
- » Right you are, Glen. I've modified the screen shot to show...
- » There are a number of things that should be included "out...
- » Hi I think it's a good thing that users as far as C4 is...
- » OK, I'm ready to trade my soul to upgrade. No, really,...

Clarion Magazine

Aussie DevCon: Charles Edmonds' Pro-Series

by Geoff Robinson

Published 2006-06-06

You would think that the "Fun With Capesoft" session would be a hard act to follow, but David Beggs did it with aplomb. Dave is an affable vet who, as well as looking after people's animals, writes farming software using Clarion. He also has various utilities like an image shrinker (to reduce the size of images for emailing etc) and a DOS Printer to allow you to print from DOS to a windows device and various icon sets - more on that later. Judging from this presentation, if he ever tired of cows and software he might consider becoming a comedian.

David was presenting three programs written by Charles Edmonds: ProScan, ProImage, and the Clarion Code Editor.

ProScan

ProScan allows you to easily add scanning to your applications. It is for C6 and uses/requires ImageEx 3.5 imaging library. ProScan is provided as source code - no black boxes. You get an APP and DCT as well as templates. It also ships with help files for the end user interface.

You can scan in multi-page documents to a multi-page tiff file and you have control over

whether to show the TWAIN scanning software that came with your scanner or not. ProScan does all the file handling automatically for you.

You can scan images into either a blob field on your current record, or in a separate blob file, or just save the images to disk.

There were various options to insert pages into the middle of an existing tiff file and so on.

David did not have a scanner at the conference and so had pre-recorded a scanning session using Instant Demo. His commentary on this had us in stitches as he said "and now this pause is while I realised I had not put the document in the feeder" and so on.

All in all it looked a very easy product to use to get scanning into your Clarion app.

ProImage

The next product was ProImage. This also uses ImageEx and allows your program to do all the usual imaging things like rotation, zooming, cropping and so on. David used some humorous photos including a rather erect kangaroo - a patient that David claimed was "obviously glad to see me." This product is still in development but we got a good idea of what it can do.

The two products are also designed to work together: ProScan can scan an image and send it straight to ProImage for editing.

The Clarion Code Editor

The final of these three programs was The Clarion Code Editor (CCE) based on the open source editing component Scintilla. CCE is a modern full 32 bit MDI editor and looked to have similar capabilities to that promised for C7

After DevCon I asked David some questions on CCE and he redirected me to its author, so much of what follows is taken from my correspondence with Charles Edmonds - and is more detailed than actually presented by David.

CCE features code-folding, color syntax highlighting (on Clarion source files, template

code, projects - as well as other file formats such as HTML and CSS), the ability to click a Clarion reserve word and open the Clarion help for that word, unlimited "UnDo" during your editing session, and a lots of other features that we all expect from a modern editor.

You can open multiple files at the same time - either from a file dialog box or by simply dragging them from Windows Explorer and dropping them onto the CCE. Each open file is represented by a "tab" (presented with the "tab buttons" of the PowerOffice PowerToolbar) above the regular button toolbar and you can navigate between open files by simply clicking the tab of your choice.

The state of each document is preserved - so wherever you leave your cursor positioned (including highlighted text, etc.), it is the same when you come back to that file.

The CCE was developed as a tool to help the delivery of the source code that comes with Clarion ProSeries tools. Charles has been quite clever here using Clarion's ability to `INCLUDE(filename, section)` to pull his production code into the TXA jumpstarts provided with the ProSeries tools.

This means that you can import his TXA and customize the look and feel of the APP to your own needs - but when Charles ships a new version of the ProSeries tool you will not have to lose your changes by importing a new TXA or do a lot of extra work by having to import code changes one at a time, because Charles' code will be in the include files.

The include files are in typical Clarion syntax example:

```
SECTION( 'Global Data' )
xxxx
xxx
SECTION( 'Global Includes' )
xxxx
xxx
```

Charles uses multiple files (one for the global - one for each of his procedures) so that changing code in one of them does not trigger Clarion to recompile all the modules/procedures.

Each file has a file extension of ".CCI" (the CCI stands for Clarion Code Include).

So a typical program that uses the ProSeries technology might have files like this:

```
MyApp.CCI
Main.CCI
BrowseCustomers.CCI
UpdateCustomers.CCI
```

Each CCI file could have multiple include sections in it such as:

```
SECTION( 'Local Data' )
xxxx
xxx
SECTION( 'Event:OpenWindow' )
xxxx
xxx
SECTION( '?ButtonSave' )
xxxx
xxx
```

When *any* of the CCI files are selected for the app, then the Clarion Code Editor will automatically figure out how many there are for the entire app and build a dynamic navigation system for them (using the Power Office XPTaskpanel).

So you might see the final result in the XPTaskpanel with the headers labeled as:

```
Globals
Main
BrowseCustomers
UpdateCustomers
```

When you click a header, it expands to display the sections in that file. So for example if you click **Globals** you might see:

```
Globals
[+]  Global Data
[-]  Global Includes
Main
BrowseCustomers
UpdateCustomers
```

... you get the idea.

Clicking on any header or task in the XPTaskpanel will open you to that section of code.

CCE actually cuts that section of code out and makes a separate document from it for you to edit, the idea being that this prevents you from accidentally editing the wrong section of code as you navigate up and down in your source file.

If you click the header for a file - you see a complete view of the entire file, whereas if you click a section then you see only that section. Changes made in a section are instantly merged into the full view in the header.

One area of CCE that Charles is very enthusiastic about is the fact that you are able to jump between your source code for different procedures and even the globals without the drilling up and down that we are used to in the standard Clarion editor.

I must admit after DevCon I was somewhat unclear as to how CCE interacted with the Clarion IDE. This was probably because David just had some screenshots rather than a working program to demonstrate. Charles explained the interaction thus:

> There is a global extension template that links the CCE to your program.
>
> If you are using CCE *with* the ProSeries tools....
>
> > The ProSeries tools have their own templates that have buttons on them that will open the CCE directly to the specific area of code in the include file.
> >
> > That CODE embed template also has a checkbox that you can uncheck so that the code would *not* be included. This is so the developer can write their own code and have it preserved when the next version of the ProSeries code is released.
>
> If you are using CCE *without* the ProSeries tools....
>
> > When you add the CCE global template, several common "stub out" points automatically provided such as:
> >
> > ```
> > Global Data
> > Global Includes
> > Global Procedures
> > Global Routines
> > (etc)
> > ```

When you add a CCE extension template to any procedure, this also provides common "stub out" points such as:

```
Local Data
EVENT:OpenWindow
EVENT:CloseWindow
EVENT:Notify
(etc)
```

This means that you can simply add the CCE templates and go instantly into CCE and begin coding for your app….

The sections defined in the templates automatically appear in the CCE for that application in (again using the XPTaskpanel).

You can also go into any embed point in your program and insert a new CODE template for the CCE. It will default to the name of the insertion point, but you can override this as you see fit.

When you click the button on the CODE template, you will be taken to the CCE (if you leave it open - it will just bring the CCE to the top window) and you will be looking at a new empty embed point that is already linked to your application.

The idea is to allow you to customize your "favorite" embed points and use them as a jumpstart when you first add the CCE template to a procedure.

Charles is also looking at making it easy for you to add automatic embed points for 3rd party tools (such as NetTalk). These would also be customizable to just the ones you want, the goal being to make you fast and productive with your coding.

New life for embeds

Charles says "Of course using include files has been with us since the DOS days, but with the Clarion Code Editor we have breathed new life into the technology."

There is a lot in the CCE, and Charles also mentioned that he is designing the CCE with an "Open Widget Interface" so that a series of tools can be developed that will literally "plug in" to the toolbars of the CCE. Things like "Message box generators", "Popup Menu Generators", etc. Charles envisages that some of these will be commercial "Clarion Micro Products" (his own as well as other third party vendors) while others will be free (user contributed).

There will be a published specification on how to exactly create the Clarion Widgets and make them appear on the CCE toolbars - as well as details on how they can interact with the CCE.

Charles mentioned that there are already developers working on CCE Widgets and he expects more will come along once they see what they can do. He also hopes that this "open standard" will encourage the Clarion Development community to get behind the CCE and develop more tools that interact with it.

Asked the "when" question, Charles responded with the familiar "soon!" response.

Just joking!!

He actually said they were getting very close to shipping and his plans are to release the CCE and an update to his existing ProScan product at the same time (the ProScan update will be moved over into the new CCE distribution format).

Then he will be releasing Clarion ProImage soon afterwards (it is mostly done and just needs to be migrated into the new format as well as having the docs completed).

All customers who purchased the advanced bundle of ProScan/ProImage will score a free copy of the Clarion Code Editor.

In summing up, the Clarion Code Editor aims to provide a robust development tool that not only allows developers who use the ProSeries tools to benefit from seamless integration of upgrades, but also gives other Clarion developers a modern editor for use with Clarion today.

The Axialis icon editor

The next part of David's session was a demonstration of the Axialis icon editor.

I think it was Tony York who mentioned that there seemed to be a competition between David's d-Icons and Sue Pichotta's Ace Icons to see who could supply the most icons in a set - running to many, many hundreds of icons.

Dave mentioned that he had done a system for a client and then their legal people insisted that he sign a form saying that all the artwork was his own. He said "Well no it is not mine - but I have the necessary license to use it." This was not good enough for the lawyers though, so Dave had to create his own icons. In this demonstration David showed how he created them using Axialis Icon Workshop.

David described Axialis as "The Clarion of icon design" showing how you could use "templates" and create icons with just a few mouse clicks. In fact Dave got sick of doing all the mouse clicks and so wrote a Clarion app to automate the process!

> **Editor's note:** This is the last installment in our Aussie DevCon coverage, slightly delayed due to Geoff getting further information on the Pro Series, as noted above. For a report on the conference wrapup, see the Day 3 report, Part 1.

Geoff Robinson lives near the beach in Melbourne, Australia, and is an active member of the local Clarion User Group. His company, Vitesse Information Systems, specializes in software for local government. Geoff was impressed by Clarion back in the DOS days and grabbed the early betas of Clarion for Windows when they first became available; he has been using Clarion as his primary development environment ever since. When not in front of a computer Geoff enjoys listening to music, singing bass in a local choir, and spending time with his three young children.

Reader Comments

Add a comment

Clarion Magazine

PrintWindow Provides Alternative To Standard Reporting

by Dave Harms

Published 2006-06-28

New third party vendors appear on the Clarion scene with regularity. Some flare and fade, others settle into a niche with one or two products. A few march steadily upward into the top tier of Clarion third party vendors.

Argentina's Jorge A. Lavera looks like a man on a march. His company, Huenuleufu Development, has eight products in its lineup, and has recently hired on two developers to help with third party and custom development.

Huenuleufu's product line includes: DigitChanger for global numeric picture management; FinalStep, which globally changes program cosmetics; FullRecord for database audit trails; NeatMessage, a `MESSAGE()` replacement, WindowID, for identifying a window's application and procedure name at runtime; TaskControl (Lite and Pro), which terminates CPU-hogging programs automatically; and the subject of this review, PrintWindow, which creates line-drawn reports that duplicate window contents.

But what is it?

In essence, PrintWindow is a replacement for the Windows standard PrintScrn functionality, rather than a replacement for standard reports. But PrintWindow is much more sophisticated than PrintScrn. If you press the PrintScrn key, Windows simply copies the contents of the entire display area (including multiple monitors, if present) into the clipboard; Alt-PrintScrn will copy just the current application frame. In both cases, PrintScrn creates a bitmap of the display.

PrintWindow, however, creates a Windows MetaFile (WMF) version of your screen, containing descriptions of text and graphics, rather than a simple screen dump bitmap. Among other things this means that PrintWindow takes much less memory than a screen dump, and the WMF files can also be scaled up or down without image degradation.

Once PrintWindow creates a WMF based on your screen, it pops the file into a Clarion report. And although a line drawing isn't as faithful a representation as a bitmap (at its native resolution), it's still very useful. Consider the example in Figures 1 and 2, taken from the PrintWindow help.

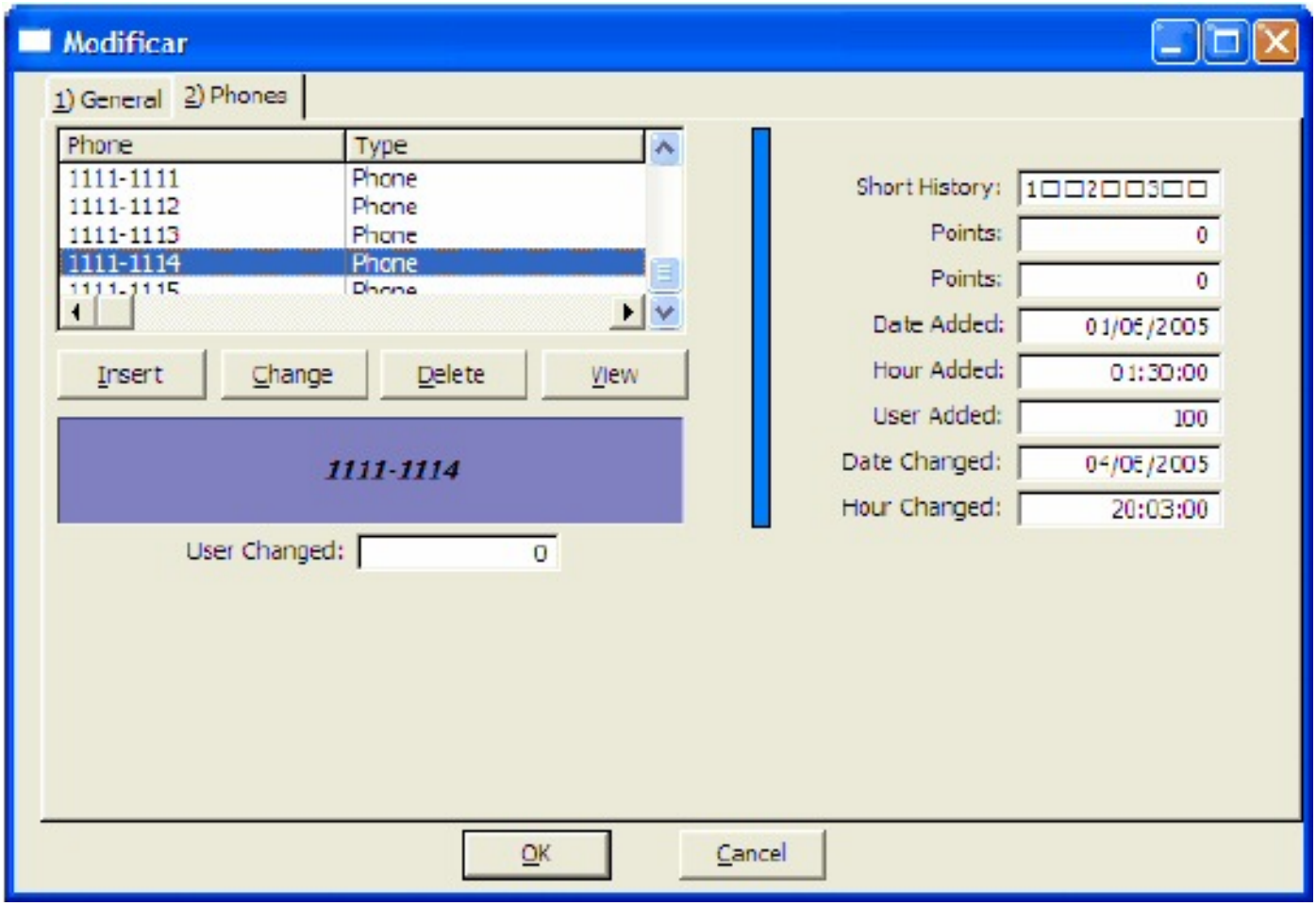


Figure 1. A window with a listbox and fields

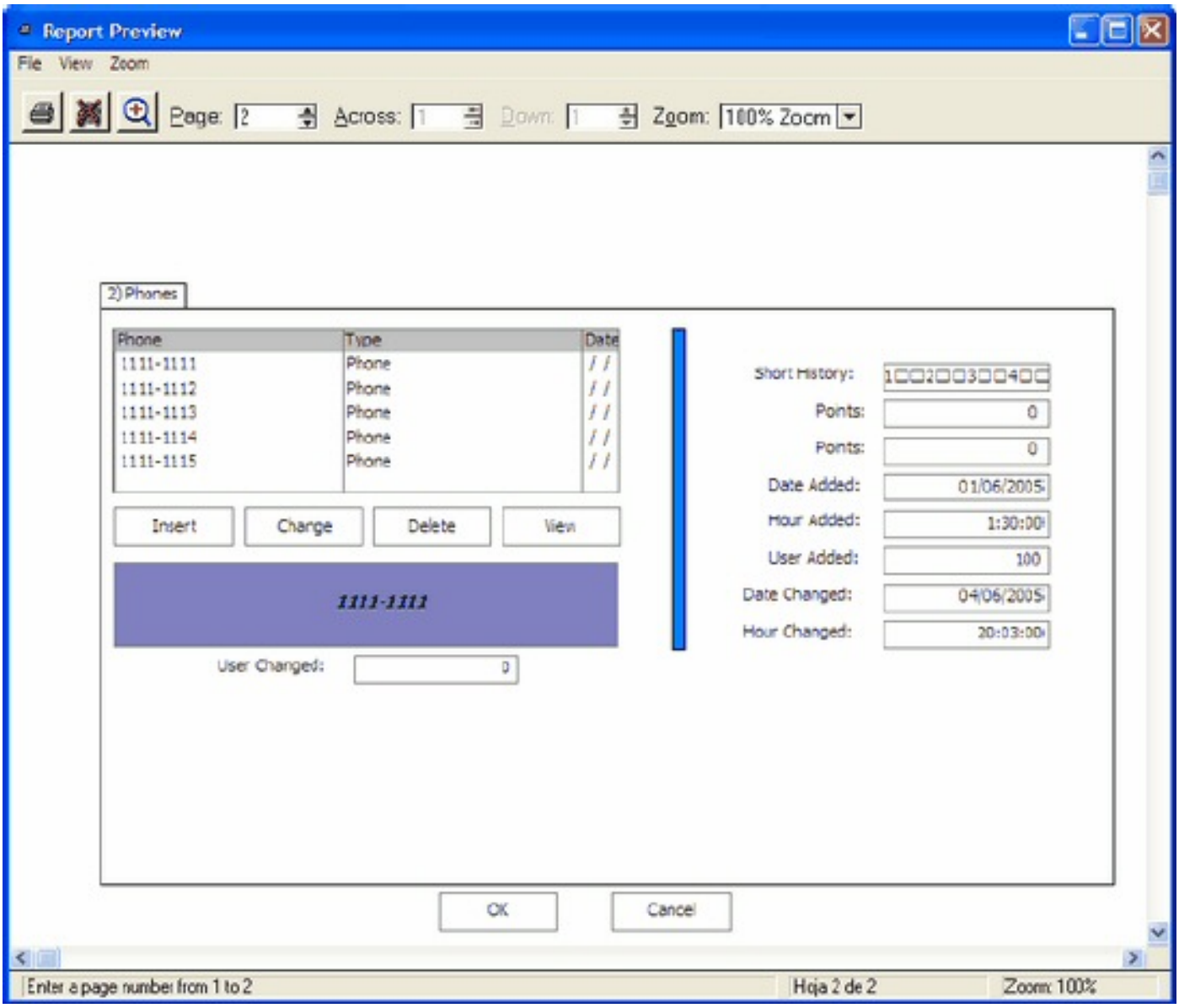


Figure 2. A PrintWindow report based on the window in Figure 1.

In Figure 2, PrintWindow has accurately reproduced the layout and data of the window in Figure 1, using simple line drawings in place of fancy bevels and shading. It would be enormously difficult to reproduce this report using standard Clarion reporting techniques.

PrintWindow draws its own representations of many Clarion controls. The main exceptions are text boxes, which are used natively (and which you can set to automatically expand to show all data), and RTF controls, but only in Clarion 6 (they are emulated in 5.5). Drawn objects such as lines, ellipses, boxes are also native.

Basic installation

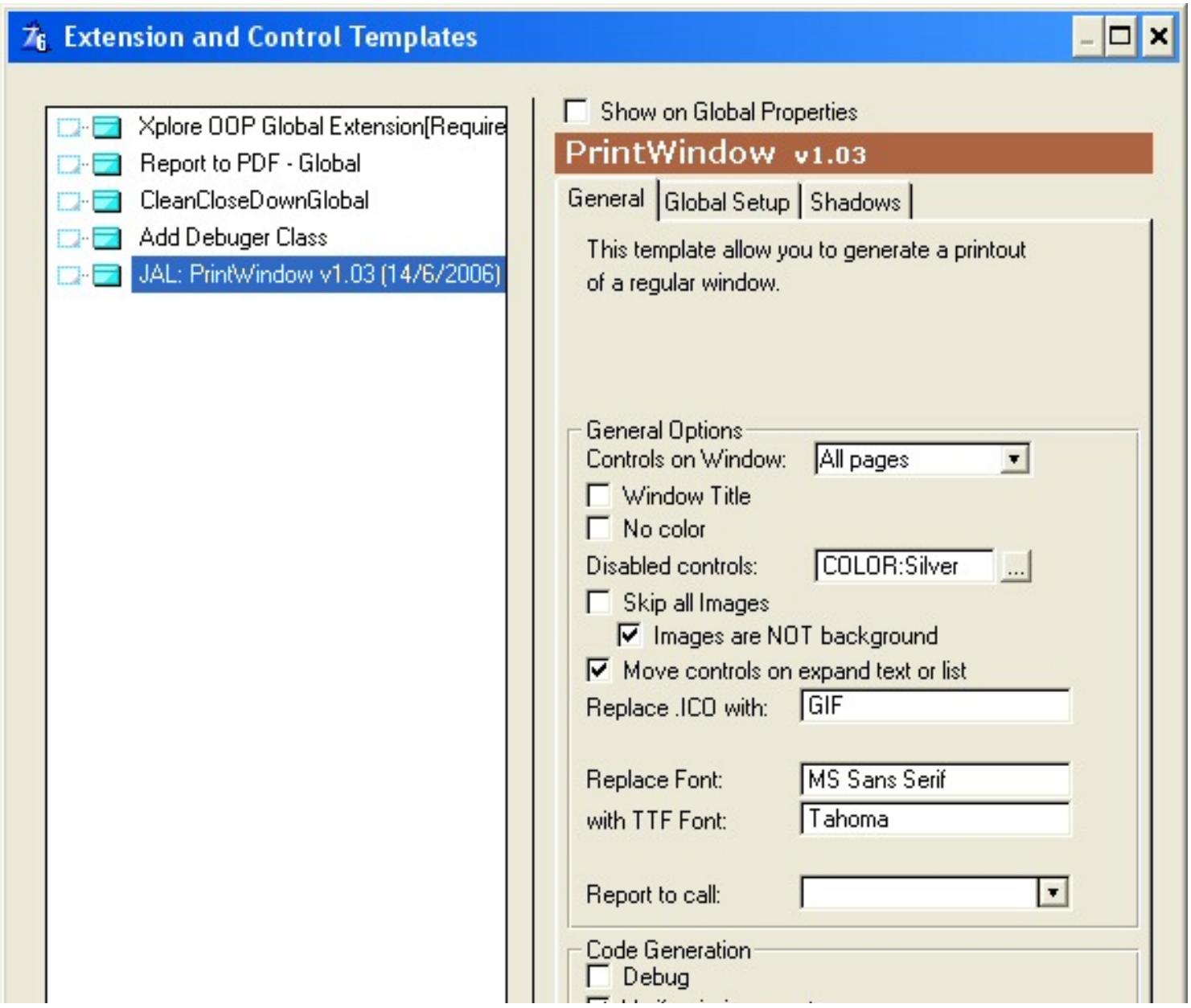
Installation went smoothly, and required a serial number as well as a length maintenance key,

both of which I simply copied/pasted into the install program. The install program also registered the necessary templates.

PrintWindow comes with a set of four global extension templates, one of which you must attach to your application. Two are for legacy apps; two are for ABC. In either case, you have a choice of a global extension that adds PrintWindow support to *every* window in your application, or a global extension that adds the basic capability to your app, after which you have to populate a control template on the windows where you want PrintWindow support.

> **NOTE:** There are also several local extensions which for some reason, on my machine, appear in the global extension list. You use these if you want to create custom reports to receive the PrintWindow WMF.

I chose the cascading version of the template, so I wouldn't have to add any control templates. Figure 3 shows the template's General options.



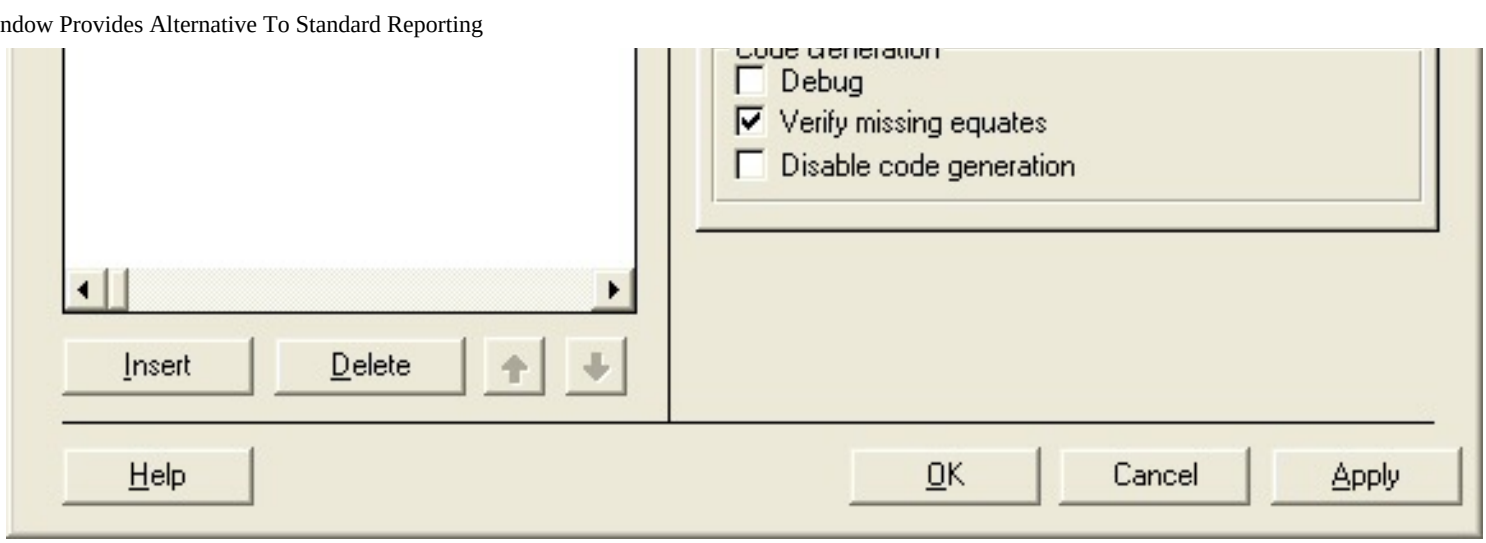


Figure 3. PrintWindow cascading template General tab

Note the drop box near the bottom, which is where you select PrintWindow's report procedure. I decided to go with one of the standard reports, which I imported from one of the TXAs in the Clarion 3rdparty\examples\PrintWindow directory.

I found two issues with the report I imported from the TXA. One, as noted in the Help file, is that I still had to add one file, any file, to the report's schematic. In C6, the template is set to use a Memory source, but the standard template prompts still require a file in the schematic, even though it isn't used. That isn't PrintWindow's fault – it's a Clarion template handling issue.

The other issue is that I ran into a compile error – the equate `?Detail` could not be found. I corrected this by going to the report's extensions list, choosing the Create a Report extension, and setting the Detail band field to `?Detail1`, as shown in Figure 4. I reported this to Jorge, who quickly fixed the proble for release 1.04.

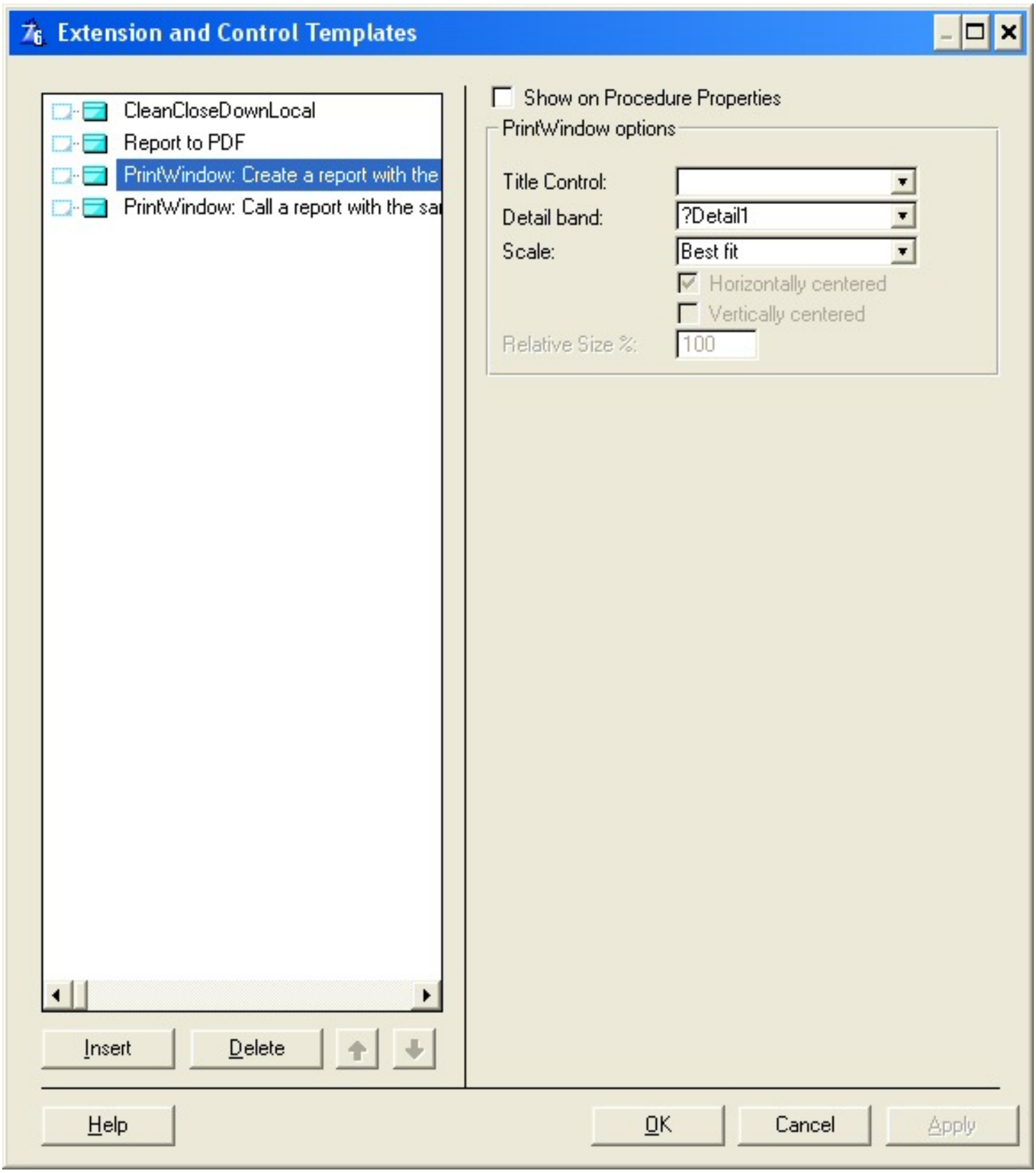


Figure 4. Setting the correct detail band.

Once I had the report procedure in place, I specified that procedure on the global extension's General tab.

If you look back at Figure 3, you'll see a checkbox in the template options called Verify missing equates. This is a generation-time check to ensure that all window controls which PrintWindow might need to deal with have use variables. The first time I generated source with the PrintWindow extension added, the template stopped on a radio button without a USE attribute. I corrected this (and several more), and I was off to the races.

I tried PrintWindow out on a number of procedures, with good results. In particular, I liked PrintWindow's ability to print out sheets with one tab per page. This is a nice way to get a more complete view of form data. You don't necessarily want to do this with browses, however. If you're using tabs in the standard Clarion way for different sort orders, you'll simply get multiple pages of exactly the same data, only with different tab headings on each page. That's because there's only one browse, which is actually contained in the window, with the sheet sitting behind it.

Even if you've chosen the cascading global extension, you can set reporting options for individual procedures, since the global template populates an extension on each procedure. Just go to the procedure's extension list and select the PrintWindow extension. Figure 5 shows the options.

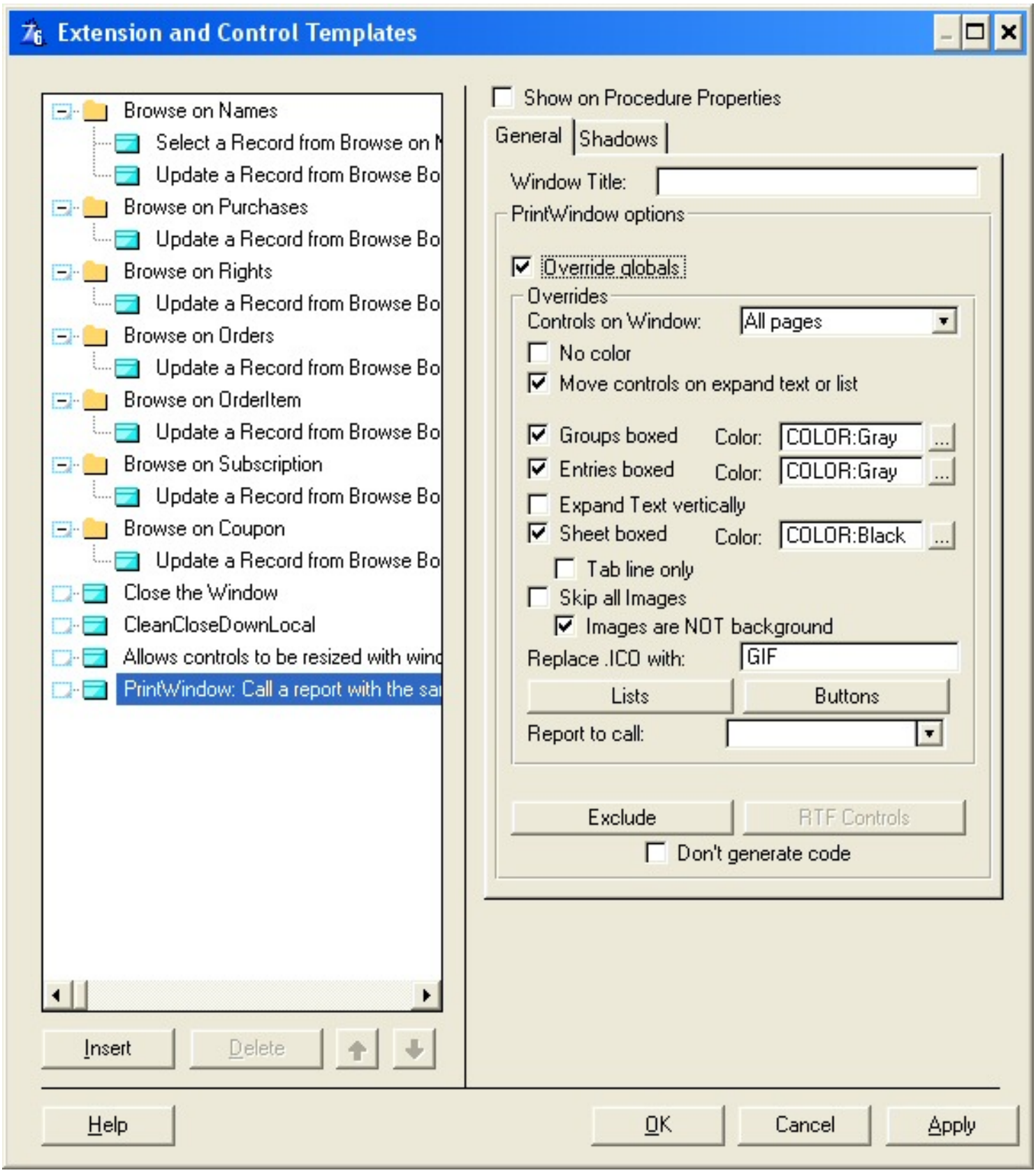


Figure 5. Procedure-specific options, with Override Globals selected

As noted in the manual, PrintWindow is not exactly a substitute for regular reports. In particular, PrintWindow's default settings only capture the browse data the user can see, beginning in column one (which means that if you scroll the browse display to the right, you'll

probably see data that won't show up in the PrintWindow report). You can, however, tell PrintWindow to display all columns, not just the visible ones – the list box will be expanded as needed, even to multiple pages. And if you are using a file-loaded browse, you can also tell PrintWindow to display all rows of data, not just what shows on screen. This works because in a file-loaded browse all of the data is in the browse's queue; in a page-loaded browse (Clarion's default configuration for browses) only a page of data is in the browse at once. So if you stick with file loaded browses (using suitable filters to keep set sizes down) you can effectively use PrintWindow as reporting tool. Add in a tool that lets you change the format of a list box by hiding, rearranging, or reformatting columns, like Icetips Xplore, and you have a fairly sophisticated end user reporting tool.

PrintWindow comes with a number of example apps, which are installed in Clarion's 3rdparty\examples\PrintWindow directory.

Other PrintWindow features include:

- Print scrolled windows that don't fit on screen
- Change appearance of controls
- Color or black and white
- Much smaller reports than when using bitmap screen capture
- Screens can be scaled larger with good results
- Override window title
- Disabled controls shown as silver (can be changed)
- Sophisticated tab handling
- Background images
- Option to skip printing images
- Font substitution (e.g. Tahoma for MS Sans Serif)
- Option for boxed groups (e.g. when no background image)
- Option for boxed entry and text fields (e.g. when no background image)
- Option to show selection bar
- Many color options
- Option to skip buttons
- Option to specify individual controls not to be printed
- Replacement of icons (.ICO) with .GIF, as reports do not support icons
- Works with your regular Clarion report previewer (whichever one you happen to have)

I've probably missed a few features – there's a lot to PrintWindow, and the documentation is extensive.

For me, the shining example of PrintWindow's usefulness is in conjunction with calendars and schedules. Figures 6 and 7, taken from the manual, show PrintWindow's version of one of Comsoft's calendar windows.

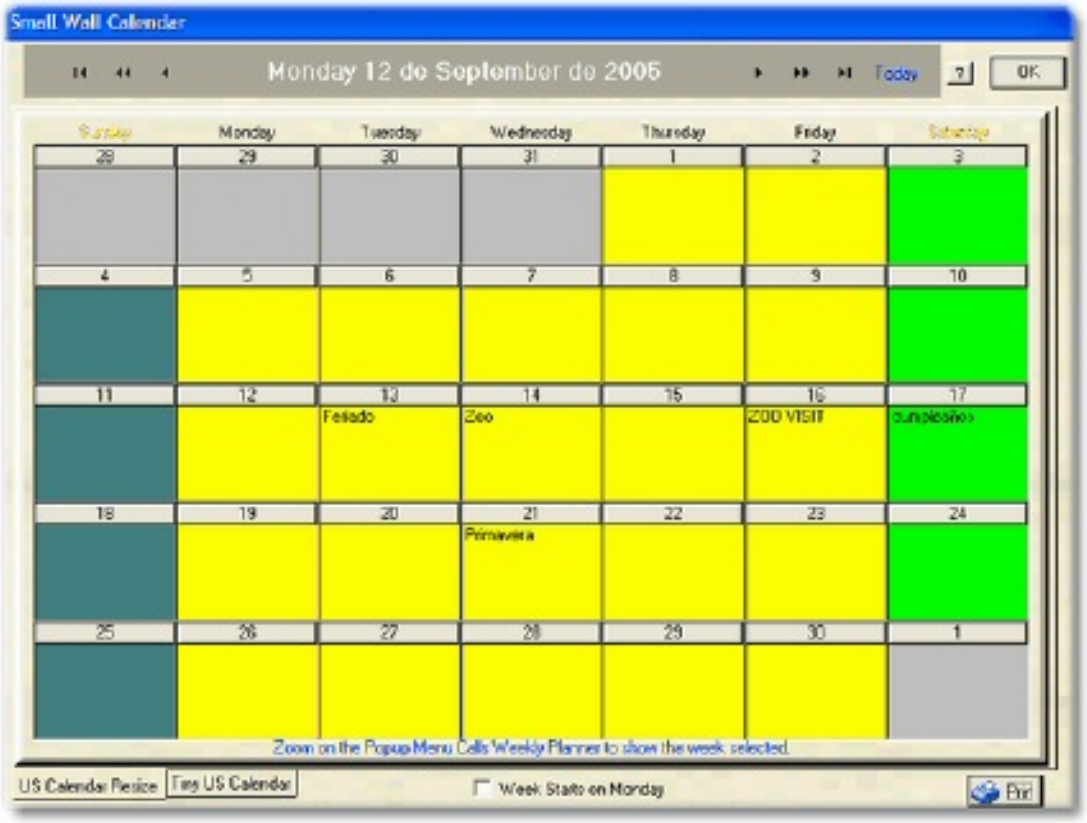


Figure 6. A Comsoft calendar

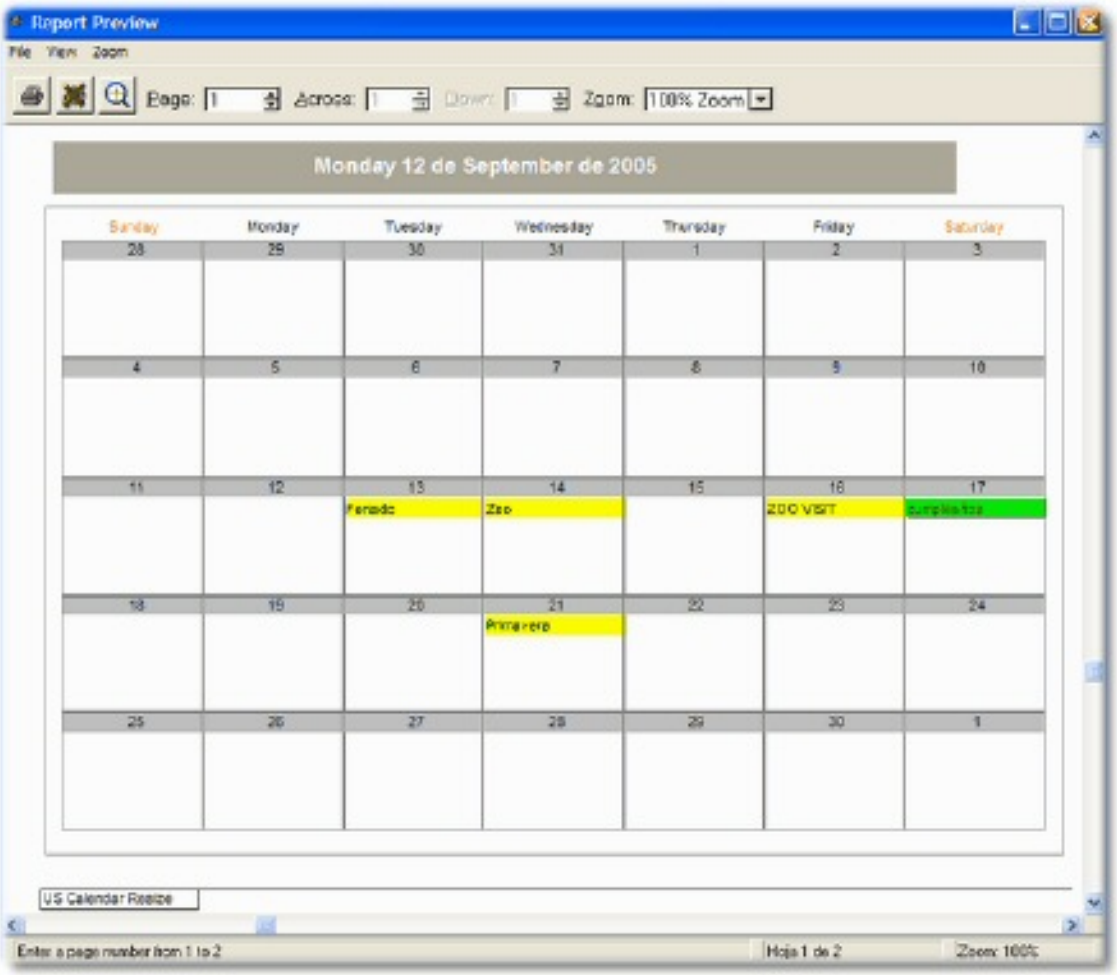


Figure 7. The PrintWindow version of the calendar

Note that the PrintWindow report has been set to skip all buttons, the bottom checkbox, and the blue message at the bottom.

It's hard enough to create something like a calendar in Clarion, where you can draw on the screen at will. To create a report version is far more work; happily, PrintWindow excels at just this kind of task.

The only problem I had with PrintWindow (other than the incorrect default detail equate mentioned above) is the appearance of what seem to be drop shadows on empty browse fields. I reported this to Jorge, and about a day later he indicated this was also fixed for release 1.04.

PrintWindow comes with a comprehensive and well-organized 146 page PDF manual, which is also available online. Although you can get PrintWindow working with just a quick skim of the installation section, there are enough options and configuration possibilities that you really will want to read the manual.

PrintWindow isn't a replacement for standard reporting, although for some quickie reports it

comes close, and even surpasses standard reporting. If you need to duplicate any part of your screen's appearance in a report, however, PrintWindow will rapidly pay for itself.

PrintWindow is avialable from Huenuleufu Development, and can be purchased at ClarionShop for $89. A demo is also available.

Rating: 

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 produces the Planet Clarion podcast, which he co-hosts with Andrew Guidroz II.

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

More June articles shortly, and a conference note

Direct link

Posted Tuesday, June 20, 2006 by Dave Harms

I've been quite busy getting the new Tips book together, so I'm a little behind on the regular articles, but that will be corrected in short order. Topics coming this month (and

into early July) include:

- using a single browse for multiple lookups
- storing business rules with the in-memory driver
- modifying the Non-Related Lookup template
- handling StarDates
- a nifty way to protect global variables
- giving your Clarion app that "glassy" look
- optical character recognition

I'll also be at the O'Reilly Open Source Convention (OSCON 2006), July 26-28, in Portland, OR, wearing my press hat. I'll be paying special attention to open source databases, but if there's a particular session you're interested in, let me know and I'll do what I can to cover it.

Proof copy of Tips book ordered

Direct link

Posted Tuesday, June 20, 2006 by Dave Harms

I've uploaded the cover and contents to the printer, and I should have a proof copy back early next week. That means we're still on schedule to ship around the end of the month, at which time the price goes up! Subscribers will pay $69.95, all others $79.95. Pre-purchase your copy now for $64.95!

This is the biggest tips book yet, clocking in at a monster 700 pages (including frontmatter and index).

ZomeAlarm 6.5 causing Clarion problems

Direct link

Posted Tuesday, June 20, 2006 by Dave Harms

There have been a numbef of reports in the newsgroups of ZoneAlarm, particularly version 6.5, wreaking havoc with Clarion 6. If you're having strange compilation problems, such as source changes not showing up in your EXE, or endless compiling, or even embeditor weirdness, and you're running ZoneAlarm, then ZA may be the cause. And you may have to uninstall ZA 6.5 and go back to 6.1, which apparently doesn't mess with C6.

Notebook upgrades and IDE to USB

Direct link

Posted Friday, June 09, 2006 by Dave Harms

Some time ago Jeff Slarve (I'm pretty sure it was Jeff) told me about the Sabrent IDE to USB adapter. This is an inexpensive device that consists of a drive power supply, and a cord with an IDE adapter at one end and a USB plug at the other. Presto, instant USB hard drive! Works like a charm, and at USB 2.0 speeds is quite snappy. I have a couple of hard drives in cheap cases, and I use the Sabrent interface (along with BeyondCompare) to do offsite backups.

Today I'm upgrading the hard drive in my Compaq Presario 2100, so I'm using the notebook drive adapter for the first time (my older version of the interface has a separate card for notebook drives, instead of a two-sided plug). I swapped the drives, installed Windows on the new drive, and got everything working. Now I'm copying across the rest of the data; the old notebook drive is sitting on my desk, attached to the Sabrent interface.

This is one piece of equipment I wouldn't want to be without. A SATA version is also available.

Benjamin Krajmalnik on PostgreSQL

Direct link

Posted Thursday, June 08, 2006 by Dave Harms

Benjamin Krajmalnik has kindly given me permission to repost his newsgroup message on PostgreSQL. I've had a long-standing interest in PostgreSQL, and if I had ClarionMag to do over, I'd look hard at this database over MySQL.

One thing to keep in mind - Benjamin's post is from the perspective of using PostgreSQL with mainly non-Clarion tools.

> I will chime in with my comments based on my experience, since I use all three. Right now, I am really liking PostgreSQL over all of the others. Feature-wise, it is superior to all the others (at least in my opinion). Their richness in data structures is amazing. One nice feature (and, please guys, do not yell at me!), is the support for arrays. While in most applications I would not use these, there are some where it comes in quite handy.
>
> For example, in our network monitoring back end, I have array structures which hold data in buckets. These buckets are dynamically aggregated. To create the structures which I am using in the other SQL flavors would have been, well, painful. Using PostgreSQL made it a breeze.
>
> Performance-wise, I am not having problems (and I insert over a million rows per day). It is very efficient in its data storage. It supports table partitionning, which is another really cool feature.
>
> PostgreSQL is an object relational database. This means you can derive objects inside the datanbase. So, in my case, I created a table structure to hold raw numeric data. I then derived 12 tables from it (one for each month). All access to the "collection" of tables is done via the parent table. A trigger routes the record insertion to the correct partition. If I want to query data, I query the parent table, and it cascades into the child tables. When the data retention is over, I simply truncate the partition. This is instantaneous - no need to manage huge indices, since the indices are actually declared on the child tables. Imagine deleting 30-50 million rows per month in the other SQL flavors. I had to do it with SQL srerver, and had to write a stored procedure which broke the deletion set up into manageable sets so as to not overflow the transaction log.

Another nice thing is that it is very good with resources. Forget the MS SQL Server paradign where it gbbles up all of your memory.

Their licensing is the best - a BSD license - which means you can do anything you want with it (unlike GPL). Support in the mailing lists is amazing, and their are various vendors providing commercial support (Command Prompt, EnterpriseDB, Pervasive, GreenPlum).

Cons - a little bit more difficult to back up the database over, say, SQL Server. There are various options available. pg_dump, which creates a consistent backup of the database, but requires performing some additional tasks with sequences after a restore, since the sequences may not reflect the maximum valued in the database.

MySQL - well, I have played with it mostly i the 3.x series and some on the 4.x series. Getting better, but my biggest concern lies with their transactional database engine, InnoDB, which now belongs to Oracle. So, in short, Oracle has MySQL by the you know what. While I found it to be great for simple websites, etc., I was still a bit wary to use it in an Enterprise environment.

SQL Server - not a bad product overall, but very expensive licensing wise (unless you use MSDE or SQL Express, in which case you have database size limitations). My biggest beef is the fact that it eats up resources like a bat out of hell. All 3 are good options.

My commercial applicatios right now run under SQL Server (since they are 3rd party add-ons to a package runing on SQL Server). My new projects are PostgreSQL based. I have not done too much with Clarion (the current project is PHP absed), but the next step in this project will also require a Windows client for some of the added features for which a Web UI is not the best.