Clarion Magazine

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

Clarion News

Read 18 Clarion news items.

Articles

DevCon Subscription Special Extended

October 5 2011

The 2011 DevCon has wrapped up, but we still have some content to add including photos and more commentary. Meanwhile, we're extending the DevCon subscription special until Wednesday, October 12 2011. Get a two year Silver subscription or renewal and save $75.

DevCon Day 1 Summary

October 8 2011

An overview of everything that happened on Day 1, with photos.

DevCon Day 2 Summary

October 9 2011

An overview of everything that happened on Day 2, with photos.

DevCon Day 3 Summary

October 10 2011

An overview of everything that happened on Day 3, with photos.

The Clarion Connection to Big Data

October 27 2011

David Harms takes a closer look at Bob Foreman's presentation on Big Data and its curious Clarion connection.

No, You Can't Return From Here

October 28 2011

Mike Hanson explains this new (in Clarion 7) error message and provides a solution.

Offshore Clarion Development Services (Advertisement)

October 28 2011

CCS Technologies is a Software Solutions Company established in 1980, having its development center based out of InfoPark, Cochin, India. CCS main focus areas are in Clarion services, SAP services, SharePoint Solutions and Mobile Applications.

Customizing Apps With Wizard Tabs

October 29 2011

Every client has a different need, it seems, and creating apps flexible enough to accommodate those different needs is tricky. Rising to the challenge, Steve Parker demonstrates the many faces of Wizard Tabs.

A Windows 7 Alt-Lockup Fix For Clarion 6, Part 3

October 29 2011

There's still more to be said about the Alt key fix for Clarion 6. Carl Barnes tweaks some code and handles one more lockup condition.

Rationalizing Parameter Lists With Groups

October 30 2011

Procedure prototypes with many parameters present opportunities for typographical errors, or as Steve Parker calls them, "typotunities". The answer: Groups!

Passing Named Groups Across Threads

October 31 2011

Having showed how to pass groups as parameters, Dr. Parker takes on a new challenge: passing groups to (and from) threads, via START().

Clarion News

Mega Bundle - 18.2GB, Over 28,000 Icons

1stLogoDesign's Mega Bundle includes 9 individual collections. Some of them contain various themes, if you break it up you get 36 collections including the various themes. $891 worth of icons for $99.

Posted October 6 2011 (permanent link)

Hunspell Clarion template and classes

Hunspell is the spell checker of LibreOffice, OpenOffice.org and Mozilla Firefox 3 & Thunderbird, amongst others. Full code and template provided. This checker is free to use. Theres a readme in the zip. The files need to be copied to the relevant directories in Clarion. No installer as yet. Changes in ver 1.0.1: Added Project statment to template to add lib to project; Fix a possible problem of drawing squiggly lines on the wrong window.

Posted October 6 2011 (permanent link)

AdiColorManager for Legacy

In the wake of Noyantis' price decrease this spring, some rushed too fast to buy AdiColorManager, not taking time to read the "ABC templates only" note. As it turned out, Bjarne had a couple of legacy applications that could use a refreshment, so he made templates for the Clarion template chain. It might be some functionality that is not duplicated in the Clarion templates, but none he has missed so far.

Posted October 6 2011 (permanent link)

CalendarPro Wrapper Template 2.11

Version 2.11 of Noyantis' CalendarPro Wrapper Template is now available for download. This update includes: Codejock v15.1.0 -> v15.1.3 compatibility added; Global Resources can now be inherited; C55 Support Code Removed; Moving / Resizing of events can now be blocked on an individual event basis via the 'EventMoving' and 'EventResizing' derived methods; Custom Colours facility added to DatePicker; New method - SetEventProperty; New DatePicker Method: 'ClearCustomColours; New DatePicker Method: 'SetCustomColours; Bug fix - CalcDate/Time now handles spaces in Locales; Bug fix - Red Flash corrected in 'DeleteAllEvents; Bug fix - Custom Colours not displaying correctly; Bug fix - Keystroke events incorrectly calculated. The update is free to all users who have an Active Maintenance Plan in place, or costs $47.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of Noyantis' web site.

Posted October 6 2011 (permanent link)

DockingPane Wrapper Template 2.02

Version 2.02 of Noyantis' DockingPane Wrapper Template is now available for download. This update includes: Codejock v15.1.0 -> v15.1.3 compatibility added; New Child Window Manager Class added; Global Resources can now be inherited; AppFrame Extension added; New Option: 'Disable Window

Close via Esc Key; New method - DeleteChildWindows; New method - RedrawChildWindows; New method - StartChildWindows. The update is free to all users who have an Active Maintenance Plan in place, or costs $47.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of Noyantis' web site.

Posted October 6 2011 (permanent link)

ChartPro Wrapper Template 1.03

Version 1.03 of Noyantis' ChartPro Wrapper Template is now available for download. This update includes: Codejock v15.1.0 -> v15.1.3 compatibility added; Global Resources can now be inherited; C55 Support Code Removed; New method - PrintChart; New method - PrintChartToDC; New method - PrintChartViaFile; New method - PrintPreview; New Chart Printing procedure added to demo app; Bug fix - Keystroke events incorrectly calculated. The update is free to all users who have an Active Maintenance Plan in place, or costs $47.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of Noyantis' web site.

Posted October 6 2011 (permanent link)

PropertyGrid Wrapper Template 2.02

Version 2.02 of Noyantis' PropertyGrid Wrapper Template is now available for download. This update includes: Codejock v15.1.0 -> v15.1.3 compatibility added; Global Resources can now be inherited; AppFrame Extension added; C55 Support Code Removed; Default Category and Item Height properties added enabling all heights to be set in advance of definition; New method - GetItemCount; Method removed - AddCtrl; Method removed - GetCtrlObj; Method removed - SetAllItemsProperty; Bug fix - Keystroke events incorrectly calculated. The update is free to all users who have an Active Maintenance Plan in place, or costs $47.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of Noyantis' web site.

Posted October 6 2011 (permanent link)

EasyQuickBooks Integrator Demo

Work is progressing on the EasyQuickBooks Integrator, which provides easy-to-use components for QuickBooks development, facilitating tasks such as adding, updating or retrieving customer information, vendor information, employee information, transactions etc. A demo is now available. Requires QuickBooks Integrator V5 ActiveX/COM Edition, ActiveX Controls and ASP COM Library (32-bit & 64-bit), Version 5.0.4230.

Posted October 6 2011 (permanent link)

ShortcutBar Wrapper Template 2.04

Version 2.04 of Noyantis' ShortcutBar Wrapper Template is now available for download. This update includes: Codejock v15.1.0 -> v15.1.3 compatibility added; Global Resources can now be inherited; AppFrame Extension enhancements (Initialize / Resize / Position). The update is free to all users who have an Active Maintenance Plan in place, or costs $47.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of Noyantis' web site.

Posted October 6 2011 (permanent link)

SuiteControls Wrapper Template 1.06

Version 1.06 of Noyantis' SuiteControls Wrapper Template is now available for download. This update includes: Codejock v15.1.0 -> v15.1.3 compatibility added; Global Resources can now be inherited; Support for Visual Theme Resource files added. (Office 2007, 2010 + Win7); Support for Visual Studio 2010 Theme added; C55 Support Code Removed; Bug fix - Keystroke events incorrectly calculated. The update is free to all users who have an Active Maintenance Plan in place, or costs $47.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of Noyantis' web site.

Posted October 6 2011 (permanent link)

amazingGUI 1.2.4.0

amazingGUI version 1.2.4.0 has been released. Fixed in this version: When using an application via remote desktop, going from the remote desktop to the local desktop and returning to the remote desktop made the application to hang (PT.242). Users with an active suscription will recieve installation keys.

Posted October 6 2011 (permanent link)

iQ-XML Updated

Changes to iQ-XML include: Enhanced and fixed documentation issues; Enhanced XML:LoadQueue function to ignore prefix/namespaces from field names. iQ-XML is a free tool for Clarion developers to add XML functionality to their applications with very little knowledge. It offers many features not found in Clarion's own XML functions. iQ-XML comes with both Parser and Writer functions. Generate an XML document from a Clarion Queue, Group structure, or using the API's. A novice user can read a complex XML document and fill a Clarion Queue easily. Navigate easily through the XML document, finding nodes and parsing only what you need.

Posted October 6 2011 (permanent link)

StrategyOnline Update

StrategyOnline recently released new builds of J-Zip and J-Skype, and overhauled the website's FAQ system.

Posted October 6 2011 (permanent link)

ClarionAddins Update

Brahn has released a new feature in the PropertyGridExtras addin: Remember Expanded State. This feature saves and restores the expanded/contracted state of categories in the property grid. This feature is "context sensitive" so for example, all button controls in the window designer can be setup so that they have a different set of expanded/contracted categories to a list control. No more hunting for properties.

Posted October 6 2011 (permanent link)

amazingGUI 1.3.0.1

amazingGUI version 1.3.0.1 has been released. Fixed in this version: Resizing problem with Entry Box, Spin Box and Text Box; MouseOver on Radio Buttons and Check Boxes - if this controls was not inside a Sheet/Tab container a black block was drawn rather than the control. Users with an active subscription will have installation keys sent to them.

Posted October 14 2011 (permanent link)

amazingGUI 1.3.0.2

amazingGUI version 1.3.0.2 has been released. Fixed in this version: In the template source code there comments that were not correctly parsed by older Clarion versions. Users with an active subscription will have installation keys sent to them.

Posted October 14 2011 (permanent link)

DMC 2.4.3.1539

DMC 2.4.3.1539 is now available free of charge to all DMC customers who have an active DMC maintenance and support plan subscription (and to all evaluation level users). Changes include:; Added two new functions during data transfers : parse (n) using a comma as separator and a pipe as separator; Added support for Table Names containing ':' in Port to SQL retrograde; Added support for a table containing a Q in Port to SQL retrograde; Added support for Port to SQL Table Key and Column names containing a ':'; Added code in TXA parsing to properly rename columns used in SetFilter(' expressions '); Added better support for code to change a procedure name when this code is in embed % BeforeProcessData; Added better support in TXA parsing when a local variable was starting with a table name; Added support to correct a c6 bug where changing a table prefix in dct editor does not always change the window controls; Changed behavior during these parse (n) functions so that if the selected 'n' does not exist in the column then an empty string is sent; Changed code to workaround a string parser pb found when an empty cstring '' is seen as having a value of ' ' by the compiler; Corrected code so as to display properly transferred data when the destination is a TPS with embedded tableprefix in column names; Changed code during TXA parsing which should eliminate more local variables been seen as table names and also speed up the processing; Corrected code when linking columns by name (data transfers) to support Access tables with Blobs; Changed code so that when doing a concat_before or after during data transferring - the concat string value is used instead of a hard coded space; Added support for table named 'event' and 'message' during a Port to SQL; Corrected a regression bug on massUpdate (stored queries would not display); Corrected a bug during Port to Sql when working on DAT Tables (mapping profile used a tps extension instead of a dat one); Corrected code to add support for a LIKE(MyColumnName) in a local data declaration during Port to SQL; Corrected code when transferring from ASCII as source so as not to SORT the Q on column 1; Corrected code when Porting to SQL when a procedure is declared as GLOBAL not to rename it as GLOBALsql; Corrected a bug during Runtime Engine using HyperFiles; Corrected a bug on Dynamic indexes during Port to SQL to not choke Clarion when importing a TXA; Corrected a bug when an SQL Table has a FLOAT to read it as a REAL; Corrected a bug in all screens which would end up with a CAPTION; Fixed a bug when cloning an Access DB to SQL when there is a MEMO-BLOB; Fixed a bug when cloning an Access DB to SQL - sql scripts were not saved to HD (path containing a dot in name ...); Fixed a bug in internal sql class on Access tables with a PK named PRIMARYKEY where DMC would read it as PK_PK_PK_PK_PK_PK_PRIMARYKEY; Corrected a bug during MySQL port to sql when grouped date and time columns where present - also touches postgre sql - the cols where not grouped and the time col was dropped; Corrected a bug during a table Clone with SQL normalization applied to remove surrounding '[' & ']' from the table name; Corrected a bug during Data Transfers on SqlAnywhere where the trunaction would return an errorcode and prohibit the transfer; Corrected a bug during SQL cloning on Descending Keys and when a PostGre SQL Table was used on schema "public".

Posted October 14 2011 (permanent link)

LANSRAD ProArchive

LANSRAD has released ProArchive, an ABC template set that makes it easy for you to add advanced record archiving and archive viewing to your existing applications - without the need to create new procedures. Records are archived into separate physical files that have the same file structure as that of the active files. Archiving can be done from a browse at the click of a button to archive the current record, or you can use a process procedure or handcode to initiate bulk record archiving. The values of auto increment fields in the active records can be preserved in the archive file copies. This makes sure that related data records stay linked to archived records. ProArchive can be used in copy only mode, or it can delete the record from the active file after the copy is complete to move records into the archive file. Both the parent record and linked child records in other files can be archived at the same time. By default the archiving process makes use of the Transaction Manager to ensure the integrity of the archive action. An option to allow records to be unarchived is included. ProArchive includes these template extensions that allow you to view or report on the archived records without the need to create new procedures. Both Single EXE and Multi-DLL demo apps are provided that show how to use ProArchive by itself, with ProPath, with FM3 and with both ProPath and FM3. There is also a "One Button Export/Import" feature that eliminates duplication of effort when configuring ProArchive for Multi-DLL apps. Limitations: 1) ProArchive is ABC Only. No support for Legacy (Clarion Templates) is provided. 2) Since ProArchive makes use of physical files to share browses and reports, currently there is no support for SQL databases. 3) ProArchive supports Clarion 8, Clarion 7 and Clarion 6. ProArchive is a full source template product - no Black Box DLLs. A demo will be available for download soon. Introductory price is $99.95 USD.

Posted October 28 2011 (permanent link)

CLARION MAGAZINE
READ. LEARN. SOLVE.

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

The ClarionMag Blog

CLARION MAGAZINE
READ. LEARN. SOLVE.

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

DevCon Subscription Special Extended

Posted October 5 2011

DevCon Subscription Special Extended!

Right now you can get a two year Silver Subscription for just $175. That's $50 off the regular two rate, and a full $75 off what you'd pay if you bought two one year silver subscriptions. **Subscribe/renew now!**

This special is set to expire on Friday, October 7.

Not sure of your subscription status? Check it here.

Article comments

BACK TO TOP

DevCon Day 1 Summary

Posted October 8 2011

Devcon photos courtesy of Dave Harms, Paul Konyk and John Morter.

Sunday Reception

The conference kicked off with a Sunday evening reception, a largely informal dinner that wrapped up with the introduction of the conference co-sponsors (Arnold Young & John Hickey, of ClarionLive!, and SoftVelocity, represented by Bob Zaunere & Co.).

Marko Golem explains the tablet giveaway to an astonished Bruce Johnson

Andy Wilton, the man in motion

Longtime FoC (Friend of Clarion) Stamos Fafalios, with Bob Zaunere

Conference organizers and presenters at a secret meeting...

See also the session notes page.

Bob Zaunere: Monday Keynote address

Notes by Steve Parker

Bob Zaunere's keynote featured a "life and times of Clarion" review in contrast with a similar history of Visual Basic, showing how Microsoft regularly introduced and then abandoned technologies (including VB6, after which all VB code had to be rewritten for .NET). Bob also mentioned the 1.5 years it had taken SV to create a program clearly showing a critical bug in Windows' MDI handling. Microsoft refused to fix the bug, instead changing the documentation to deprecate MDI. SoftVelocity has continued to support MDI.

Microsoft has made many changes in user interface technologies, from WinForms to WebForms to CompactForms to WPF and Silverlight. There have been many, many more data access layer changes (the implication being that Microsoft can't make up its mind, and/or it can't commit to its developers). In contrast, Clarion provides seamless evolution and backward compatibility with all previous version of its Windows product. Much DOS code still runs in Windows.

In Windows 8 things change again; it's a XAML-based UI (WPF and Silverlight are XAML), and the recommended data access approach is now the Entity Framework ORM. Bob emphasized Clarion's stability over the years (leaving aside the need for significant rewriting going to .NET).

See also the session notes page.

The landing party security detail (John Hickey and ArnoldY oung), holding it all together at the back of the room

SoftVelocity: Clarion and the IDE

Notes by Steve Parker

C7 was a long and hard beta process (we all know this). Diego presented a long list of changes from C7 to C8 – a very long list, too many to call C8 "just" an improved C7. Changes include locators everywhere, data pad editing improvements, new ABC functionality, disabled icons, three state checkboxes.

New in C8: PNG support (almost fully baked), but not EMF (Enhanced MetaFile) just yet. There's a new option on the GPF dialog to log more detailed information. Also on the way is caching of ABC file headers, but only for Enterprise Edition.

Diego demonstrated the new disabled image rendering versus C6, gradients, changing the order of DLLs in a multi-DLL solution.

Version control integration is improved in the next release. There are visual indicators for binary import/export. You will be able to access the Subversion menu from the Solution explorer.

The key view in the dictionary has been rearranged, so there's no longer a need to go to a different tab.

The main conference meeting room

SV has done some testing to verify that you can run Clarion apps in the cloud using SQL Azure – all you need is a different connection string. When asked about connection timeout issues, Bob Z indicated that needed further research.

There was some time for questions and answers. Can SVN checkout be done at the procedure level? Not directly, but possibly by selective export.

What about fixes to TopCopy/TopScan? Z would like to see examples of corruption.

There was some discussion about a 64 bit version of the IP server, as well as web development. ASP.NET is the next step, but PHP is also a possibility, as is some support for new Windows 8 functionality.

See also the session notes page.

David Harms: Writing code backwards

Notes by Steve Parker

Dave Harms began his session with a description of the "old way" of writing code from a specification: start with a data dictionary, wizard up some windows, accommodate specific functionality with embed code. The "new" way is to create objects that model business operations. Think "how do I want to use this code" and write what looks like pseudocode. He claims this is really different from writing embedded code (part of his "move code out of embeds to classes" thesis, actually based on a need to test).

Object-oriented programming provides better modeling capabilities than procedural code. It allows automated unit testing. The big issue is really the quantity and type of code, which is not reusable when it is in embeds.

The value of an app is in the code that models business processes. Isolate that business logic by moving the code to classes.

Dave concluded by reiterating that we should think "how will I use the code" not "how will I write it". What is the highest level code I'd like to call to do X. This approach defers implementation details. Write code the way you want to use it and only then make it work.

See also the session notes page.

Bruce Johnson: The Road Not Taken

Paul provides a detailed report on the session notes page.

Shawn Mason: Optimizing SQL in Clarion

Notes by Steve Parker

So you've recently converted to SQL Server. Now what? Do you need to learn the SQL language? Shawn says No, but Yes. You don't absolutely need to learn it. But you should.

Knowledge gives you flexibility and opportunity to optimize.

As of SQL Server 2008 there are Date and Time types – use them. Don't use LONGs for dates and times. You can have RI in the dictionary or in the database, but don't have them in both. You should move RI to the database.

Browse: You need a primary key on all sort orders (at the end of the sort fields list). To refresh a browse use BRWx.ResetQueue(Reset:Queue). To reset a browse use ResetFromFile. Filtered locators work best. Use PROP:SQL where possible in place of processes (e.g. mass deletes). For mass updates it depends on the nature of the update.

Shawn defined a cursor as a temporary buffer for a set of records.

Clarion does 80% of what you need for SQL just fine as is.

For processes/reports, if you use a field in a computation but it's not populated it must be a hot field.

More definitions:

A SQL view is a query that Clarion treats as a table (or file, to use the old terminology).

Inner join: retrieve all parent records that have children.

Outer join: retrieve all parent records whether or not they have children.

Clarion's table schema defaults to left outer join but provides an "inner" checkbox for those situations where you want to filter based on the presence of child records.

Move your dictionary autonumbering to back end unique identity. It's faster and more stable. But Shawn doesn't recommend using GUIDs unless absolutely necessary for external reasons, because of the way indexes are structured. GUIDs can incur performance penalties because the indexes aren't naturally sequential.

Consider server side triggers and stored procedures. Also consider IMDD for lookup tables in heavily transactional apps. When doing bulk deletes or inserts, if you're affecting more than 7-10% of the records in a table you'll be better off dropping indexes and recreating them after the bulk operation.

Use PROP:SQLFilter to filter views.

In SQL Server 2005 Microsoft introduced Common Table Expressions (CTE) which can be used for recursive queries or non-permanent views.

There is a Merge statement in T-SQL that can make merging tables much easier.

You can improve your SQL server environment by regular profiling and making index changes as needed. Get a good defragger and use it.

See also the session notes page.

Fun with CapeSoft

The CapeSoft boys gave away the first of two iPads on Monday.

Bruce Johnson explains the rules of "rock, paper scissors" while Geoff Thomson looks on.

Dave Harms goes out early.

The winner, Michael Tabakin!

See also the session notes page.

Article comments

by Jonathan Gellie on October 11 2011 (comment link)

I enjoyed the pics and also the valuable notes in SQL. Thanks

by Dave Harms on October 12 2011 (comment link)

You're most welcome, Jonathan.

BACK TO TOP

DevCon Day 2 Summary

Posted October 9 2011

Devcon photos courtesy of Dave Harms, Paul Konyk and John Morter.

Bob Foreman: LexisNexis and Big Data (updated October 25)

It was a real treat to have Bob Foreman at the conference. Recently Bob left SoftVelocity, after two decades of service in the Clarion cause, and joined Lexis Nexis where he works alongside former Clarion luminaries including David Bayliss, Richard Chapman, Richard Taylor and others.

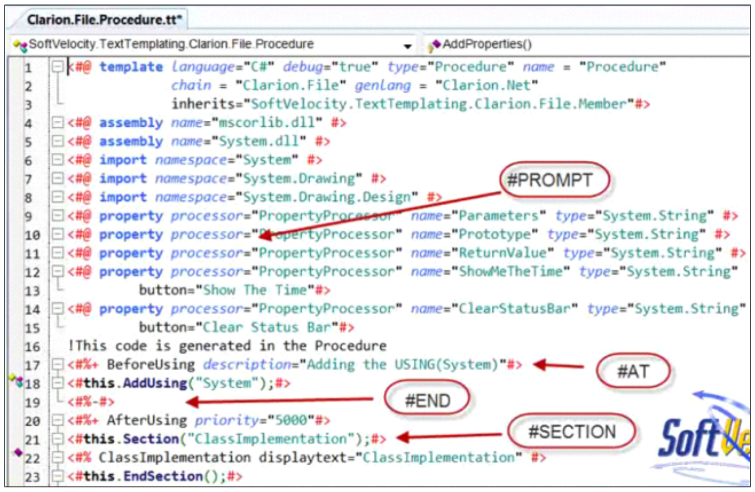
Bob Foreman and John Berger

Update: Geoff kindly reminded me that I hadn't answered the "ECL" question. Please see this article for the answer.

See also the session notes page.

SoftVelocity: Application Generation in Clarion.NET

Bob, Diego and Pierre presented a lot of information on code generation in Clarion.NET. Here's a screen shot of a portion of the Clarion.File.Procedure.tt template file.


Clarion.File.Procedure.tt*
SoftVelocity.TextTemplating.Clarion.File.Procedure
AddProperties()
#PROMPT
#AT
#END
#SECTION

```
<#@ template language="C#" debug="true" type="Procedure" name = "Procedure"
             chain = "Clarion.File" genlang = "Clarion.Net"
             inherits="SoftVelocity.TextTemplating.Clarion.File.Member"#>
<#@ assembly name="mscorlib.dll" #>
<#@ assembly name="System.dll" #>
<#@ import namespace="System" #>
<#@ import namespace="System.Drawing" #>
<#@ import namespace="System.Drawing.Design" #>
<#@ property processor="PropertyProcessor" name="Parameters" type="System.String" #>
<#@ property processor="PropertyProcessor" name="Prototype" type="System.String" #>
<#@ property processor="PropertyProcessor" name="ReturnValue" type="System.String" #>
<#@ property processor="PropertyProcessor" name="ShowMeTheTime" type="System.String"
         button="Show The Time"#>
<#@ property processor="PropertyProcessor" name="ClearStatusBar" type="System.String"
         button="Clear Status Bar"#>
!This code is generated in the Procedure
<#%+ BeforeUsing description="Adding the USING(System)"#>
<#this.AddUsing("System");#>
<#%-#>
<#%+ AfterUsing priority="5000"#>
<#this.Section("ClassImplementation");#>
<#% ClassImplementation displaytext="ClassImplementation" #>
<#this.EndSection();#>
```

This code shows how prompts and, in particular, embeds are declared in the templates. Embeds are a key part of the Clarion codegen system but are not part of the standard T4 language; this is an important enhancement.

SV presented a lot of other interesting screen shots and tidbits of information; look for an in-depth analysis of the new AppGen in ClarionMag later this month.

See also the session notes page.

Mike Hanson: Visual Design Tips & Tricks

Mike presented a ton of information related to user experience and visual design. More than just tips and techniques, this was a full-on style guide for Clarion development.

Mike Hanson

See also the session notes page.

Mark Goldberg:Clarionizing API Calls

Mark is the hand coder's hand coder. Although conversant with AppGen, he really knows how to make the Clarion language stand on its head, and provided a number of important techniques for calling non-Clarion code.

See also the session notes page.

Andy Wilton: Using COM, A Practical Approach

Paul Konyk provides a summary of Andy's presentation on the session notes page.

Andy Wilton: Jazzing up your project with Noyantis Tricks

See also the session notes page.

Article comments

BACK TO TOP

DevCon Day 3 Summary

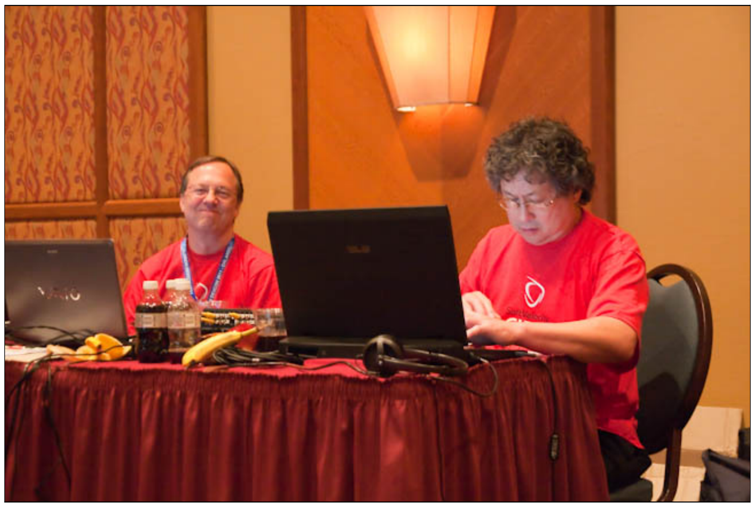
Posted October 10 2011

Devcon photos courtesy of Dave Harms, Paul Konyk and John Morter.

Marko Golem/Daniel Pavlic: Thin@ Performance

Marko and Daniel got a lot of attention at last year's DevCon; Marko is the original creator of Thin@, a thin client solution for Clarion, and he was joined a few years later by Daniel. Like a number of other third party products (notably CapeSoft's offerings) Thin@ was created to meet an in-house need, and is used to deliver a applications for several large Croation installations. As a third party product Thin@ been well received for its stability and for the level of service Marko and Daniel provide.

See also the session notes page.



Day 3 - John and Arnold, stayin' alive!

SoftVelocity: Using Clarion for Mobile and Web Development

Bob Z explained how Win32/.NET interop works in Clarion#. You can do interop in other .NET languages, but the Clarion# approach is very easy as it's similar to how you call non-Clarion Win32 DLLs. The Clarion# compiler does the work of rewriting some IL code to expose procedure (really method) calls via a normal LIB file. You declare the procedure in a Win32 map, add the lib and you're good to go.

Pierre covered LINQ for data access, using the Clarion LINQ to File provider. This led to some discussion of the data layer being generated by the new AppGen. The data layer is roughly analogous to the ABC FileManager/RelationManager classes.

Diego demonstrated calling Windows services from Clarion Win32, but using Clarion.NET's interop because the web service access code can be written much more quickly and easily in .NET than in Win32.

Bob Z closed with a discussion of MVC and a demonstration of a simple WebForms app. Although SV plans to support MVC eventually, initially their focus will be ASP.NET WebForms.

See also the session notes page.

Rick Martin: SQL ODBC Direct Class

Rick has a really useful class for doing direct SQL access via the ODBC driver. Along with retrieving data, the class displays data using a virtual list box, an underappreciated Clarion feature that came up for discussion as well.

See also the session notes page.

Mike Hanson: OOP - Interfaces, Virtual Methods, Abstract Classes and Other Scary Stuff

Mike hit the highlights of object-oriented programming patterns and practices, and illustrated a number of these in the process of solving a problem presented by Mike Gorman.

See also the session notes page.

Panel Discussion

This was a much shorter panel discussion than last year's, and focused largely on several questions from the audience on how Clarion developers can help SoftVelocity be successful. This is a question that has come up time and again, and in the opinion of Bruce and Dave, at least, it's possibly a misplaced one. Several suggestions were put forward. In his closing keynote Bob Z was asked this same question, and he suggested that referring other developers to Clarion was probably the best way.

See also the session notes page.

And another giveaway!

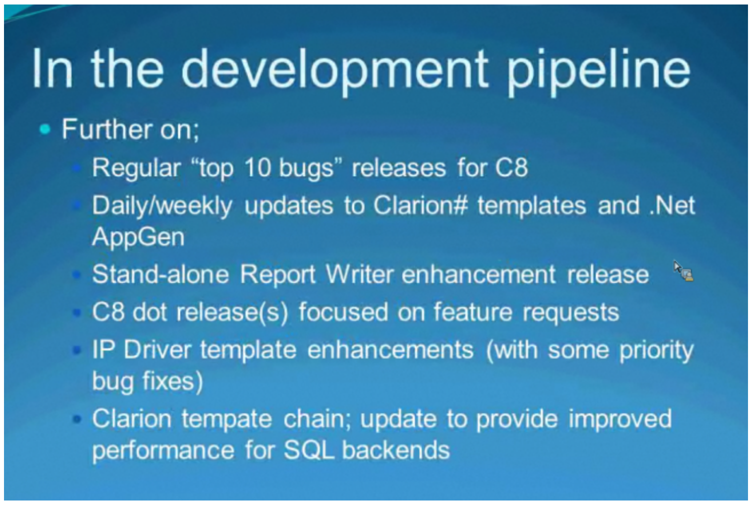
There was lots of hardware flying through the air at this conference. Andy Wilton gave away two Kindles (Paul Sivak, Lauri Soininen); CapeSoft two iPads (Michael Tabakin, Dee Witham); the Thin@ boys donated an Asus tablet computer (Ron Cookson), and SoftVelocity contributed a nicely equipped Dell i7 laptop (Eddie Sizemore)!

Bob Z presents the laptop computer to Eddie Sizemore

Robert Zaunere: Future Roadmap, Q&A

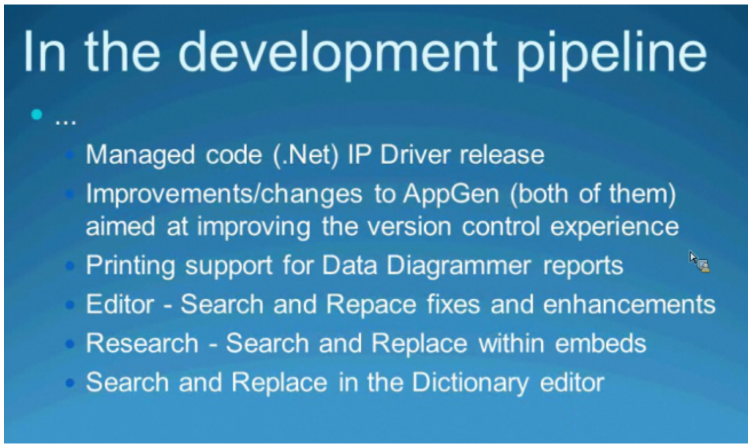
SoftVelocity president Bob Zaunere gave the closing keynote address. Among other thingshe indicated a new C8 release in around two weeks, and a first release of the Clarion.NET AppGen in four weeks. He then took a number of questions from the audience, on subjects from a possible 64 bit version to unicode to phone development. He said a trial edition is coming, along with a university edition. He also indicated SV plans a Visual Studio edition of the Clarion.NET AppGen.

A few slides from the keynote:



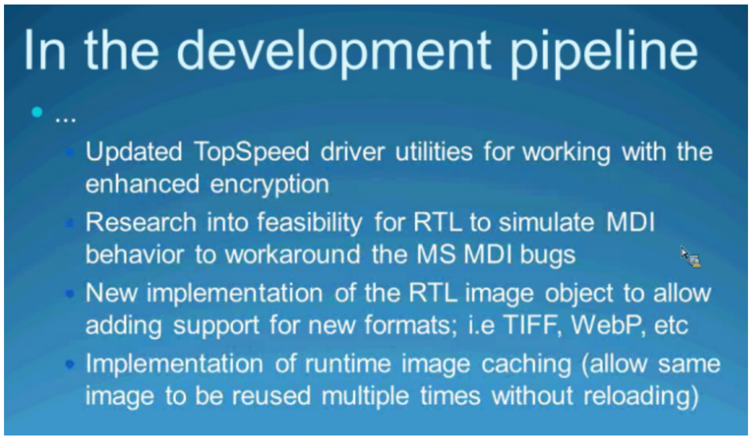
In the development pipeline

- Further on;
  - Regular "top 10 bugs" releases for C8
  - Daily/weekly updates to Clarion# templates and .Net AppGen
  - Stand-alone Report Writer enhancement release
  - C8 dot release(s) focused on feature requests
  - IP Driver template enhancements (with some priority bug fixes)
  - Clarion tempate chain; update to provide improved performance for SQL backends



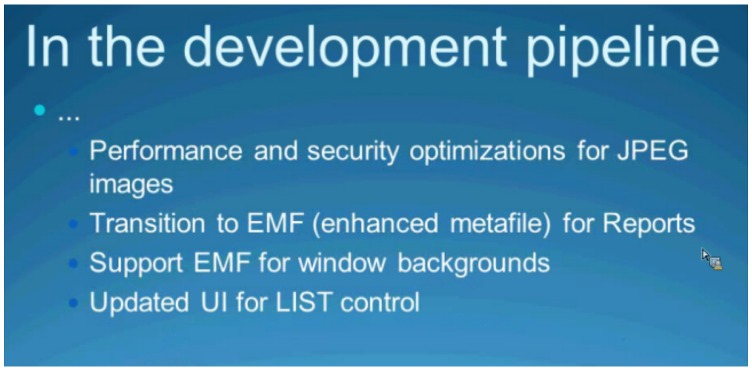
In the development pipeline

- ...
  - Managed code (.Net) IP Driver release
  - Improvements/changes to AppGen (both of them) aimed at improving the version control experience
  - Printing support for Data Diagrammer reports
  - Editor - Search and Repace fixes and enhancements
  - Research - Search and Replace within embeds
  - Search and Replace in the Dictionary editor



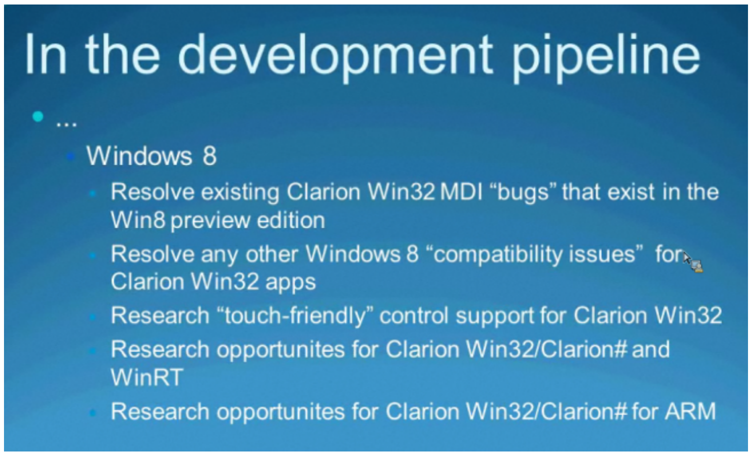
In the development pipeline

- ...
  - Updated TopSpeed driver utilities for working with the enhanced encryption
  - Research into feasibility for RTL to simulate MDI behavior to workaround the MS MDI bugs
  - New implementation of the RTL image object to allow adding support for new formats; i.e TIFF, WebP, etc
  - Implementation of runtime image caching (allow same image to be reused multiple times without reloading)



In the development pipeline

- ...
  - Performance and security optimizations for JPEG images
  - Transition to EMF (enhanced metafile) for Reports
  - Support EMF for window backgrounds
  - Updated UI for LIST control



In the development pipeline

- ...
  - Windows 8
    - Resolve existing Clarion Win32 MDI “bugs” that exist in the Win8 preview edition
    - Resolve any other Windows 8 “compatibility issues” for Clarion Win32 apps
    - Research “touch-friendly” control support for Clarion Win32
    - Research opportunites for Clarion Win32/Clarion# and WinRT
    - Research opportunites for Clarion Win32/Clarion# for ARM

New runtime features include a transition from WMF to EMF for reports, EMF window backgrounds, better image handling to allow support for new formats, possible repurposing of the (more or less defunct, in Microsoft's view) CompactForms designer for Android/iPhone work, Windows 8 compatibility and more.

See also the session notes page.

Article comments

BACK TO TOP

The Clarion Connection to Big Data

By Dave Harms

Posted October 27 2011

Bob Foreman's appearance at this year's DevCon was a special treat. Bob now works for LexisNexis, but he is best known to Clarion devs for his twenty-some years at SoftVelocity and its predecessors where his excellent training and documentation skills gained him a reputation as a really cool frood.

Bob's presentation was on Big Data, which in LexisNexis' usage is data processing that cannot be accommodated by a single machine. The answer is to split up the data *and* the processing so you can have from several to hundreds (or perhaps thousands) of machines all working away at pieces of a larger problem.

There's a significant Clarion connection here: many of the developers who created Clarion for Windows now work for LexisNexis, and specifically they're on the team that created the High Performance Computing Cluster (HPCC) product. HPCC is a competitor to the Apache Hadoop project, which is used by large sites such as Yahoo and Facebook. This year Facebook announced its Hadoop cluster was handling 30 petabytes of data.

The HPPCC team includes such former luminaries as Richard Chapman, David Bayliss, Gordon Smith and the late Nigel Hicks. Back in 2000 David Bayliss, who wrote the Clarion for Windows compiler, created what is now called Extended Control Language, or ECL.

I'm told that originally ECL stood for Extended Clarion Language, but had to be changed for obvious reasons.

Bob commented during his presentation on how familiar ECL seemed to him. I gave Bob a bit of grief about this later; I've played with HPCC a bit, and I have to say that the Clarion-like nature of ECL didn't really leap out of me, at least beyond the (to me) annoying propensity for ALL CAPS KEYWORDS.

Here's a sample program that takes a very small data set (one word of four letters) and creates every possible combination of those letters:

```
STRING Word := 'FRED' :STORED('Word');
R := RECORD
        STRING SoFar {MAXLENGTH(200)};
        STRING Rest {MAXLENGTH(200)};
    END;
Init := DATASET([{'',Word}],R);
R Pluck1(DATASET(R) infile) := FUNCTION
R TakeOne(R le, UNSIGNED1 c) := TRANSFORM
                SELF.SoFar := le.SoFar + le.Rest[c];
                SELF.Rest := le.Rest[..c-1]+le.Rest[c+1..];
// Boundary Conditions handled automatically
  END;
RETURN NORMALIZE(infile,LENGTH(LEFT.Rest),TakeOne(LEFT,COUNTER));
  END;
L := LOOP(Init,LENGTH(TRIM(Word)),Pluck1(ROWS(LEFT)));
OUTPUT(L);
```

The output is a list of all possible permutations of the word FRED, as in

```
FRED
FRDE
FERD
FEDR
FDRE
...
```

and so on.

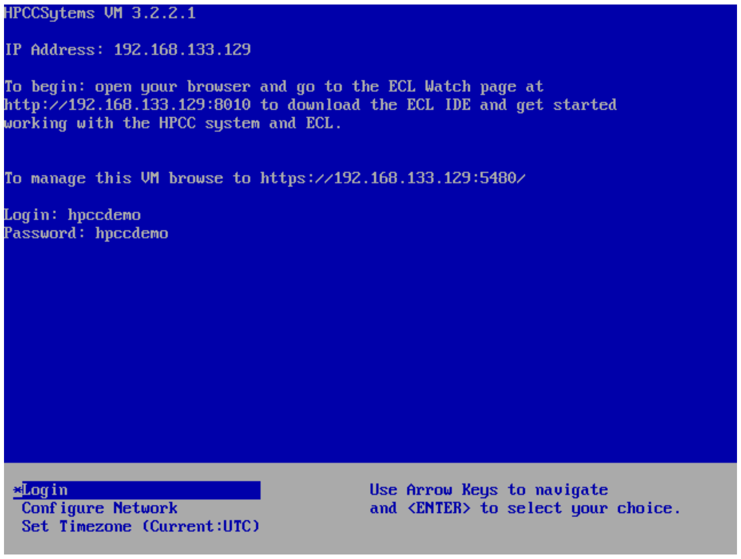
When I asked Bob specifically about the Clarionesque nature of ECL, he pointed out the approach to declaring data (e.g. record structures) and the compact, expressive nature of the language. And yes, if I squint I can definitely see some similarities there.

HPCC also has a template language, and there the statements look perhaps a bit more familiar.

Make no mistake: writing code in ECL is vastly different than writing code in Clarion. This is a different problem domain. You're not writing applications, you're processing massive amounts of data. The code just isn't going to look like normal application code. But it is very interesting to see what the big Clarion brains have been up to in the last decade or so. And who knows, even if you're not in need of massively parallel data processing now, that day could come a lot sooner than you think.

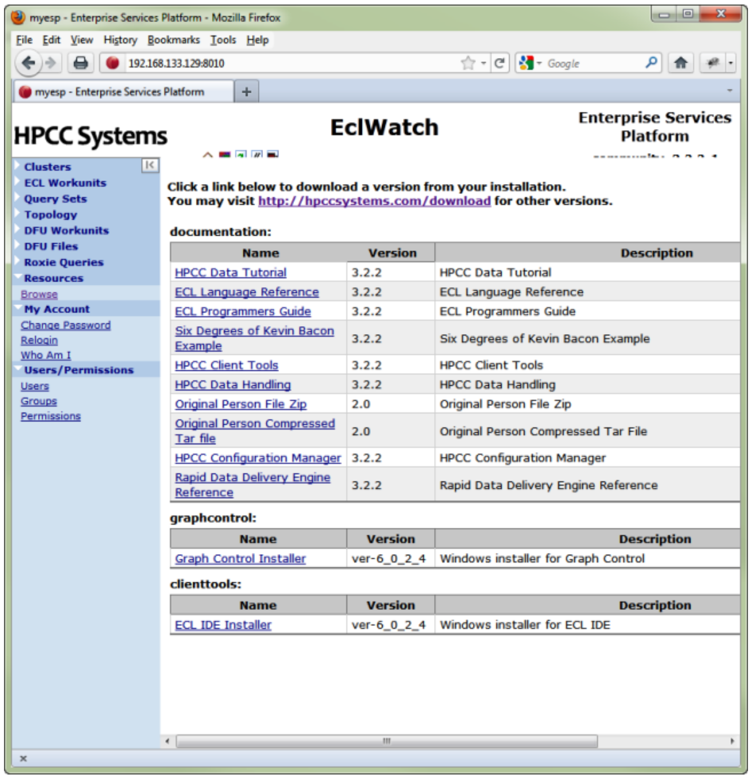
Getting started with HPCC

A free, community edition of HPCC is readily available. Source code is also available. The easiest way to get going is to download the HPCC virtual machine image and run it on your desktop. After the OS boots you'll see something like this:



```
HPCCSytems VM 3.2.2.1

IP Address: 192.168.133.129

To begin: open your browser and go to the ECL Watch page at
http://192.168.133.129:8010 to download the ECL IDE and get started
working with the HPCC system and ECL.


To manage this VM browse to https://192.168.133.129:5480/

Login: hpccdemo
Password: hpccdemo

 *Login                         Use Arrow Keys to navigate
  Configure Network             and <ENTER> to select your choice.
  Set Timezone (Current:UTC)
```

You can then use your desktop's browser to connect to a web server running in that image and download the Windows IDE.



Click a link below to download a version from your installation.
You may visit http://hpccsystems.com/download for other versions.

documentation:

Name	Version	Description
HPCC Data Tutorial	3.2.2	HPCC Data Tutorial
ECL Language Reference	3.2.2	ECL Language Reference
ECL Programmers Guide	3.2.2	ECL Programmers Guide
Six Degrees of Kevin Bacon Example	3.2.2	Six Degrees of Kevin Bacon Example
HPCC Client Tools	3.2.2	HPCC Client Tools
HPCC Data Handling	3.2.2	HPCC Data Handling
Original Person File Zip	2.0	Original Person File Zip
Original Person Compressed Tar file	2.0	Original Person Compressed Tar File
HPCC Configuration Manager	3.2.2	HPCC Configuration Manager
Rapid Data Delivery Engine Reference	3.2.2	Rapid Data Delivery Engine Reference

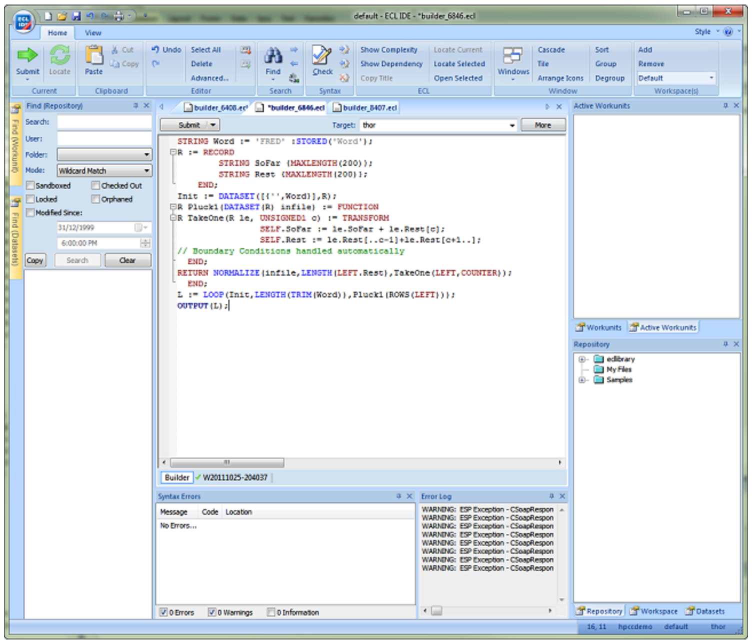
graphcontrol:

Name	Version	Description
Graph Control Installer	ver-6_0_2_4	Windows installer for Graph Control

clienttools:

Name	Version	Description
ECL IDE Installer	ver-6_0_2_4	Windows installer for ECL IDE

You write code in the IDE, on your desktop, but the code is submitted to and runs on a single node HPCC "cluster" running in the virtual machine.

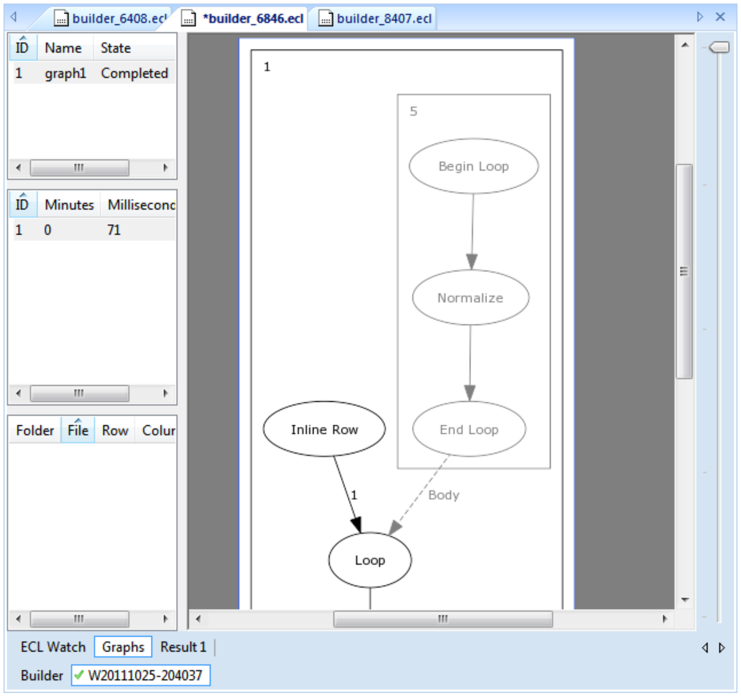


```
STRING Word := 'FRED' :STORED('Word');
R := RECORD
        STRING SoFar {MAXLENGTH(200)};
        STRING Rest {MAXLENGTH(200)};
    END;
Init := DATASET([{'',Word}],R);
R Pluck1(DATASET(R) infile) := FUNCTION
R TakeOne(R le, UNSIGNED1 c) := TRANSFORM
                SELF.SoFar := le.SoFar + le.Rest[c];
                SELF.Rest := le.Rest[..c-1]+le.Rest[c+1..];
// Boundary Conditions handled automatically
  END;
RETURN NORMALIZE(infile,LENGTH(LEFT.Rest),TakeOne(LEFT,COUNTER));
  END;
L := LOOP(Init,LENGTH(TRIM(Word)),Pluck1(ROWS(LEFT)));
OUTPUT(L);
```

```
No Errors...
```

```
WARNING:  ESP Exception - CSoapRespon
WARNING:  ESP Exception - CSoapRespon
WARNING:  ESP Exception - CSoapRespon
WARNING:  ESP Exception - CSoapRespon
WARNING:  ESP Exception - CSoapRespon
WARNING:  ESP Exception - CSoapRespon
WARNING:  ESP Exception - CSoapRespon
WARNING:  ESP Exception - CSoapRespon
```

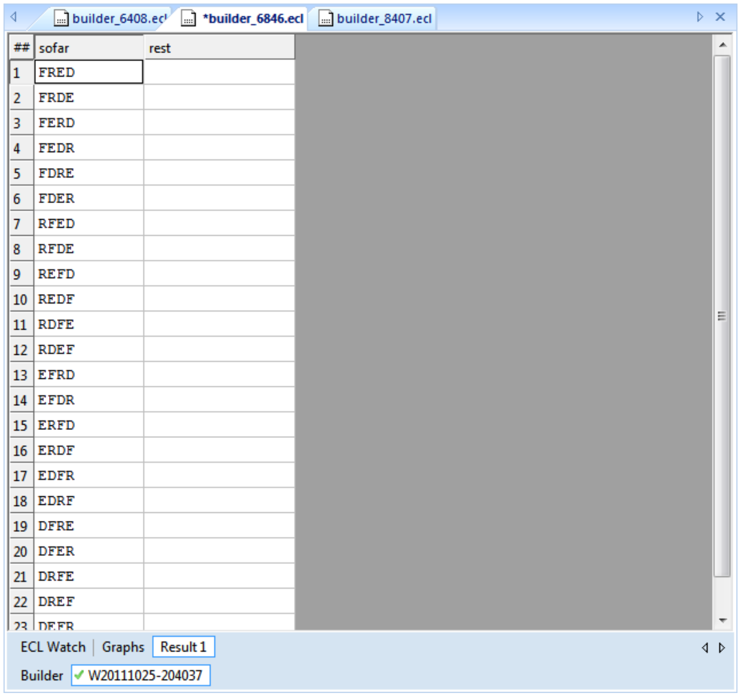
You can see graphs of program execution:



ID	Name	State
1	graph1	Completed

ID	Minutes	Millisecond
1	0	71

And you can view the results.



##	sofar	rest
1	FRED	
2	FRDE	
3	FERD	
4	FEDR	
5	FDRE	
6	FDER	
7	RFED	
8	RFDE	
9	REFD	
10	REDF	
11	RDFE	
12	RDEF	
13	EFRD	
14	EFDR	
15	ERFD	
16	ERDF	
17	EDFR	
18	EDRF	
19	DFRE	
20	DFER	
21	DRFE	
22	DREF	
23	DEFR	

As you can tell from the screen shots, this is a very different kind of development from the usual Clarion work. Unless you're already well-versed in big data and products like Hadoop, you'll probably want to read the Getting Started docs and watch the videos.

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

Article comments

BACK TO TOP

CLARION MAGAZINE
READ. LEARN. SOLVE.

No, You Can't Return From Here

By Mike Hanson

Posted October 28 2011

Have you ever encountered a "ROUTINE that can RETURN from procedure cannot be called here" compiler error? This error didn't appear in Clarion 6. Why is it showing up now?

Back in the old (pre ABC) days, it was common practice to `DO ProcedureReturn`. That routine would perform a bunch of cleanup (closing files, setting global response variables, etc.), then it would issue a `RETURN` statement to exit the procedure where it was housed.

```
MyProc              PROCEDURE
  CODE
  !Some other code
  DO ProcedureReturn

ProcedureReturn     ROUTINE
  !Clean-up
  RETURN
```

As time passed, Clarion was enhanced to support local classes (with their methods), local procedure `MAP`s. Consequently, returning from a routine became more ambiguous. Why? Let's look at the ordering of elements within a procedure:

- `PROCEDURE` statement
- Optionally, procedures defined in a local `MAP`
- Local data, including `CLASS`es
- Main code (following the `CODE` statement)
- Procedure Routines (belonging to the entire procedure)
- Any procedures declared in a local `MAP`, and any methods for locally declared `CLASS`es.

The last point is especially important. Note that each of these methods/procedures can house its own set of routines, class methods, local map procedures, etc. And those deeper local procedures can house their own routines, methods, etc. It's very fractal.

Not only can your code call down into these nested methods/procedures, but those methods/procedures at the various scoping depths can call the `ROUTINE`s of the method/procedure in which they are housed, as well as any of their own routines.

C6 supported this organization, but it never complained about the potential confusion. In C7/8 we're suddenly getting the error, "ROUTINE that can RETURN from procedure cannot be called here". Why is this now a problem? Consider this example:

```
MyProc              PROCEDURE
MyClass               CLASS
MyMethod                PROCEDURE
                      END
  CODE
  MyClass.MyMethod
  !Some other code
  DO ProcedureReturn

ProcedureReturn     ROUTINE
  !Clean-up
  RETURN

MyClass.MyMethod    PROCEDURE
  CODE
  DO ProcedureReturn  !***ERROR***!
```

MyClass.MyMethod is at a deeper scope within MyProc. When it calls the ProcedureReturn ROUTINE for MyProc, should it RETURN from MyClass.MyMethod or from MyProc entirely? Rather than decide for you, as in previous versions of Clarion, the compiler now produces the error. And this makes sense from a readability point of view also – issuing a return from within one nested level (a routine) isn't so bad; issuing that same return from several levels deep could make for some seriously cryptic code.

You can work around this by using a local variable to communicate that the greater procedure should be aborted, and/or change your methods and procedures to return an "Abort" flag and pass that up the calling chain. The approach you use will depend on your overall coding style. For example:

```
MyProc              PROCEDURE
AbortProc             BOOL(FALSE)  !***
MyClass               CLASS
MyMethod                PROCEDURE
                      END
  CODE
  MyClass.MyMethod
  IF AbortProc THEN DO ProcedureReturn.  !***
  !Some other code
  DO ProcedureReturn

ProcedureReturn     ROUTINE
  !Clean-up
  RETURN

MyClass.MyMethod    PROCEDURE
  CODE
  AbortProc = TRUE  !***
```

The best option is to never `RETURN` from a `ROUTINE`, but with so many Legacy procedures and their `ProcedureReturn` routines, that's not going to happen. We can't change generated code, but we can change how and when we call it.

Hopefully this helps to clarify why this error suddenly appeared in our previously *perfect* programs.

Mike Hanson is affiliated with BoxSoft, which produces the "Super" series of templates, distributed through Mitten Software. He has been creating add-on products for Clarion since his Public Domain Models for CPD 2.0 back in 1988. He's also written articles for every Clarion-related publication, and has spoken at numerous conferences and training seminars. If you have any questions, you can reach him via www.boxsoft.net.

Article comments

BACK TO TOP

Offshore Clarion Development Services (Advertisement)

Posted October 28 2011

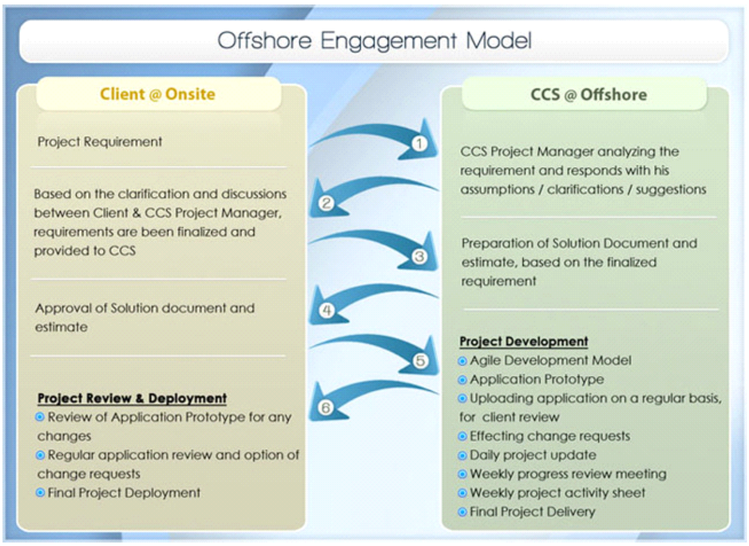
by Thomas Mathew (tmathew@ccstechnologies.in)

This is a paid advertisement

About CCS

CCS Technologies is a Software Solutions Company established in 1980, having its development center based out of InfoPark, Cochin, India. CCS main focus areas are in Clarion services, SAP services, SharePoint Solutions and Mobile Applications. CCS is working as an Offshore Development Partner for a number of clients from US, Canada, UK, Netherlands, Norway, Australia, Singapore and Middle East.

CCS Offshore Working Model



Offshore Engagement Model

Client @ Onsite

Project Requirement

Based on the clarification and discussions between Client & CCS Project Manager, requirements are been finalized and provided to CCS

Approval of Solution document and estimate

Project Review & Deployment
- Review of Application Prototype for any changes
- Regular application review and option of change requests
- Final Project Deployment

CCS @ Offshore

CCS Project Manager analyzing the requirement and responds with his assumptions / clarifications / suggestions

Preparation of Solution Document and estimate, based on the finalized requirement

Project Development
- Agile Development Model
- Application Prototype
- Uploading application on a regular basis, for client review
- Effecting change requests
- Daily project update
- Weekly progress review meeting
- Weekly project activity sheet
- Final Project Delivery

• Project Management by Client / CCS

• CCS Developers working as extended team of Client

• Directly reporting to Client Project Lead

• Directly working on Client’s machine / Develop locally and deliver

• Signing of NDA / SLA / MoU for the protection of Client IP Rights

CCS & Clarion

CCS started using Clarion, way back in 1992, with the DOS based Clarion Professional Developer, and then migrated to subsequent versions and now we are in Clarion version 8.

We are providing the following Clarion Services on an Offshore Delivery Model to our overseas clients.

Clarion Offshore Services

• Developments in Clarion 8

• Conversion from Clarion 5.x, 6.x To 7.x and 8

• Clarion.Net — Applications to run on Desktop, Web and Mobile devices

• Maintenance in Clarion Applications — (New Functionality, Reports and Bug fixing)

• Database Migration to MSSQL / Oracle / MySQL / Sybase SQL

• Development of New Templates, Maintenance in Legacy/Third-party Templates

• Clarion Integration with other applications developed in Java/PHP/.Net using Web services and SOAP protocol

Conversion from Clarion 5.x/6.x TO Clarion 8

CCS understands the demand for converting Clarion applications in 5.x / 6.x to current Clarion 7.x / 8 versions which will be compatible with compiling in 64 bit machines. CCS has expertise in converting big complex applications to Clarion 8 from Clarion 5.5.

Integration of Clarion Application with SAP ERP

One of the primary challenges of SAP implementation process has been integrating SAP with the existing Legacy system, on which the customer has spent the money and time, building complex business specific logic, to interact directly with the ERP system. If the legacy application happens to be in Clarion, the challenge is even more, since the expertise is very limited. CCS has the experience in integrating legacy Clarion applications to SAP. The integration has done using the Remote Function Call (RFC) communication facility provided by SAP.

Benefits of Outsourcing

• Drastic Reduction in Operation Costs

• Reduced Development/Delivery Timeframe

• Increase manpower without additional recruitment

• Reduced administration of Human Resources

• Possibility to take up more projects

We look forward to become your Clarion development partner and provide the required support

on your Clarion applications.

Please write to me for more details at tmathew@ccstechnologies.in

Clarion Services

• Reliable

• Innovative

• Cost effective

• On-time

• Friendly

This is a paid advertisement

BACK TO TOP

Customizing Apps With Wizard Tabs

By Steven Parker

Posted October 29 2011

Vertical market applications, I have discovered, all share one transcendent quality. That quality is that, no matter how well planned, no matter how well executed the application, the next prospective customer does "*X*" differently.

And "*X*" is always a mission critical, company specific computation. It will not do, not at all, to tell the prospect "This is the way I do it, it is not changeable." Perhaps it is the customer's method of price computation, perhaps it is how the customer computes margins. But, whatever it is, it is something that, unchanged, prevents a sale of the software.

One option for the developer is to maintain custom versions of the software, one for each customer insisting on being non-normative. Need I argue that this isn't really an option?

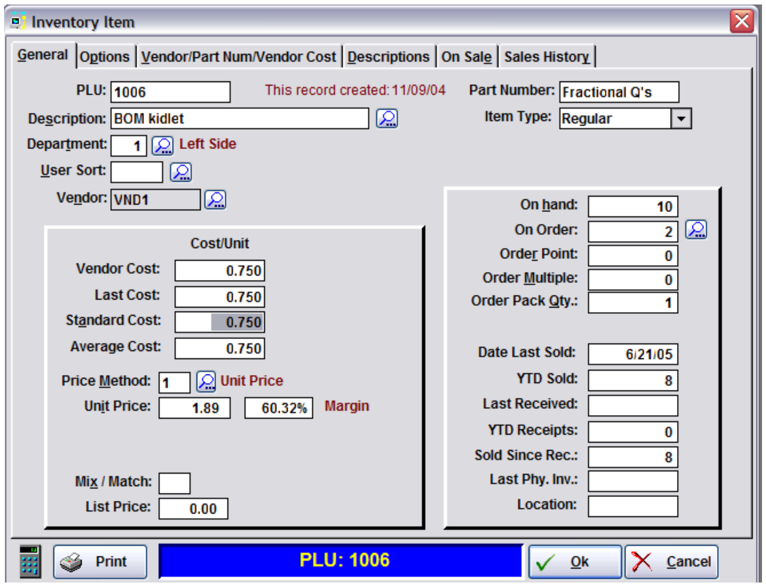
Another option is to have multiple update forms, customized on a per customer basis:

```
Case Config.SerialNumber
Of 12345            ! customer A
  UpdateForm1
Of 23456            ! customer B
  UpdateForrm2
! etc.
Else
  UpdateForm        ! standard form
End
```

To implement this tactic, for each customer with a different computation method, copy and rename the base form. Then, changing the computation becomes straightforward. But, the application bloats. And, while maintenance is not precisely a nightmare, it isn't exactly all that simple.

While this is entirely workable, it really is … inelegant (voice of experience here).

Hiding and unhiding fields based on a condition is workable. Consider a POS inventory form:


Inventory Item

General | Options | Vendor/Part Num/Vendor Cost | Descriptions | On Sale | Sales History

PLU: 1006 This record created: 11/09/04 Part Number: Fractional Q's
Description: BOM kidlet Item Type: Regular
Department: 1 Left Side
User Sort:
Vendor: VND1

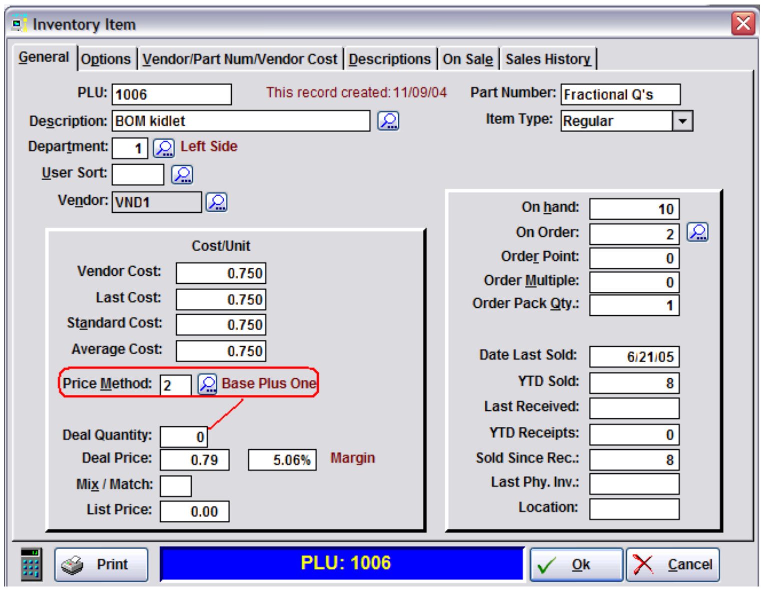
Cost/Unit
Vendor Cost: 0.750
Last Cost: 0.750
Standard Cost: 0.750
Average Cost: 0.750
Price Method: 1 Unit Price
Unit Price: 1.89 60.32% Margin
Mix / Match:
List Price: 0.00

On hand: 10
On Order: 2
Order Point: 0
Order Multiple: 0
Order Pack Qty.: 1
Date Last Sold: 6/21/05
YTD Sold: 8
Last Received:
YTD Receipts: 0
Sold Since Rec.: 8
Last Phy. Inv.:
Location:

Print | PLU: 1006 | Ok | Cancel

Figure 1. Standard inventory item

and the different data required if a price rule is not “eaches” (unit price for a single item):


Inventory Item

General | Options | Vendor/Part Num/Vendor Cost | Descriptions | On Sale | Sales History

PLU: 1006 This record created: 11/09/04 Part Number: Fractional Q's
Description: BOM kidlet Item Type: Regular
Department: 1 Left Side
User Sort:
Vendor: VND1

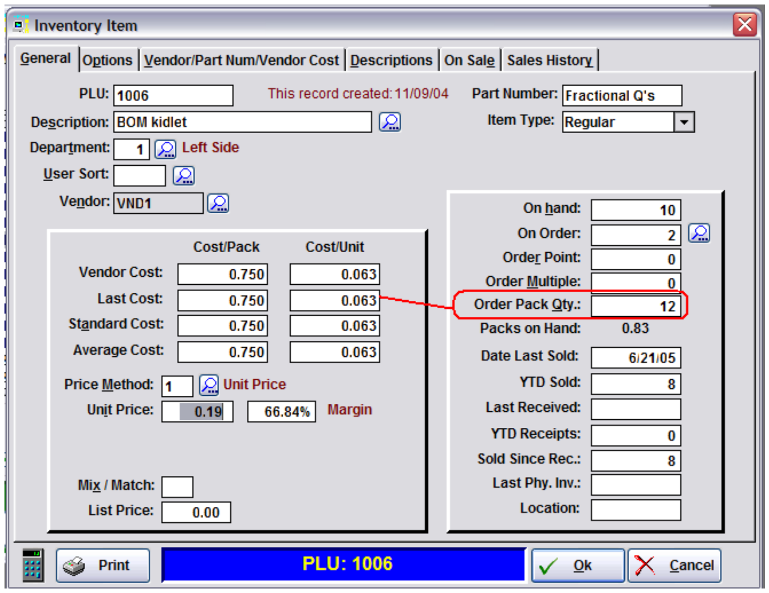
Cost/Unit
Vendor Cost: 0.750
Last Cost: 0.750
Standard Cost: 0.750
Average Cost: 0.750
Price Method: 2 Base Plus One
Deal Quantity: 0
Deal Price: 0.79 5.06% Margin
Mix / Match:
List Price: 0.00

On hand: 10
On Order: 2
Order Point: 0
Order Multiple: 0
Order Pack Qty.: 1
Date Last Sold: 6/21/05
YTD Sold: 8
Last Received:
YTD Receipts: 0
Sold Since Rec.: 8
Last Phy. Inv.:
Location:

Print | PLU: 1006 | Ok | Cancel

Figure 2. Inventory form for different price rule

Now consider the information required if the item is purchased in pack quantities but sold by the unit (e.g., by the box, not by the unit, a case of beer sold by the six pack, for example):



PLU: 1006 This record created: 11/09/04
Part Number: Fractional Q's
Description: BOM kidlet
Item Type: Regular
Department: 1 Left Side
User Sort:
Vendor: VND1

	Cost/Pack	Cost/Unit
Vendor Cost:	0.750	0.063
Last Cost:	0.750	0.063
Standard Cost:	0.750	0.063
Average Cost:	0.750	0.063

Price Method: 1 Unit Price
Unit Price: 0.19 66.84% Margin
Mix / Match:
List Price: 0.00

On hand:	10
On Order:	2
Order Point:	0
Order Multiple:	0
Order Pack Qty.:	12
Packs on Hand:	0.83
Date Last Sold:	6/21/05
YTD Sold:	8
Last Received:	
YTD Receipts:	0
Sold Since Rec.:	8
Last Phy. Inv.:	
Location:	

PLU: 1006

Figure 3. Inventory form for "pack quantities"

In Figure 2, there are a series of fields that are hidden or unhidden depending on the value of "Price Rule." Because there are 10 price rules, there is a fairly serious `CASE` structure handling the hiding and unhiding. (You don't really want to know what this window looks like in design mode.)

The change in Figure 3 is triggered by the primary vendor's selling information. If the vendor sells the item only in pack quantities and the item is sold in units, a series of fields is unhidden. The `CASE` here is somewhat simpler, only 20 or so lines of code.

Labor intensive, to be sure, and I would not relish having to make changes to these areas. However, the underlying computations are entirely standard (once the price rule, pack quantity, margin/markup method and margin/markup percent are known). There are no customer specific variations. Customers may not have unique combinations of data entry fields available based on price rule or vendor pack quantities. Because the app I am describing is a retail POS, pricing computations are standardized, customers do not have unique computations once the price rule, etc., are entered.

What if the customer *does* require a unique entry field (or fields) and unique computation (obviously, in a non-retail app)? Hiding/unhiding fields might (or might not) be a workable solution. But how do I handle the unique computation required?

A Case in Point

A fellow Clarioneer faced just such a quandary recently. In this developer's app, one screen collects information and uses it to compute a license fee.

The app was initially developed for a specific municipality. Several other municipalities became interested. But, it turns out that other municipalities use both different bases and different formulas for computing the license fee. In fact, the license fee basis and computation is a matter of local law (specs written by politicians). Therefore, the odds on two customers computing on the same data and using the same computation, while better than "slim to none," is low.

The developer built the app based on

```
(square footage) * (per foot amount)
```

which seems entirely reasonable. But he discovered that another municipality used:

```
(number of electrical outlets) * (per outlet fee)
```

and another might well base the license fee on the length of the county supervisor's nose or on the square root of the Golden Ratio. Who knows? These algorithms were written by politicians.

A Solution

It seemed to me that there would be a finite number of computation methods. Each could be identified by a numeric designation.

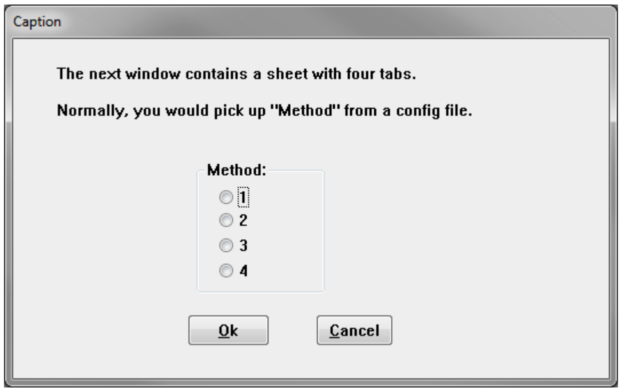
It seems easy enough to me to pass this enumerator to a Source template procedure as a parameter. That parameter could trigger the correct computation in a `CASE`:

```
Case pMethod
Of 1
  pFee = pBaseValue * .25
Of 2
  pFee = pBaseValue * 23.75
Of 3
  pFee = pBaseValue * 112
Of 4
  pFee = pBaseValue * 8
End
```

`pBaseValue` is the data entered by the user (also a parameter passed into the calculating procedure). It could the the square footage or whatever the particular municipality uses.

The `Method` variable, passed as pMethod, is a datum that I would store in a configuration file and read at an appropriate time. If the computation is a straight multiplication, as in the code snippet above, it too could be stored in a configuration file. In this case, multiple municipalities could use the same Method but different multipliers, reducing the nodes in the `CASE` tree.

In the demo app accompanying this article, I set up a window to collect `Method` (a real user would never see such a window). This allows iterative testing with different values for the variable:



The next window contains a sheet with four tabs.

Normally, you would pick up "Method" from a config file.

Figure 4. Window to Replace Reading Configuration File

All that remains is presenting the user with...

The Appropriate Prompt and Entry Field

Because the number of different prompts and data types I might need is unknown, I don't want to go down the path of overlaying, hiding and unhiding multiple prompts and entry fields.

There is a way of handling the need for multiple, differential displays (as discussed in "Secondary Forms (Part 1)"). That method is tabs, one per Method, dynamically selected at run time:

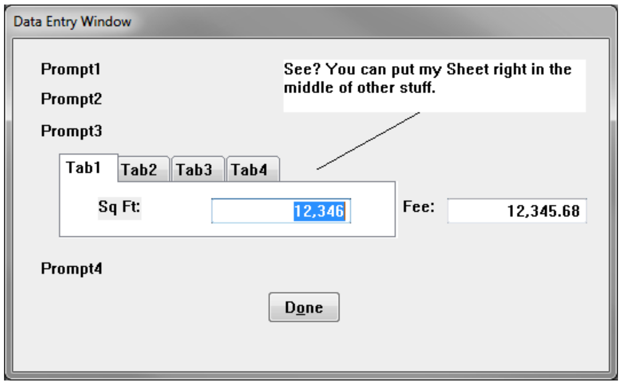


Figure 5. Tabs for Showing Different Entry Fields:

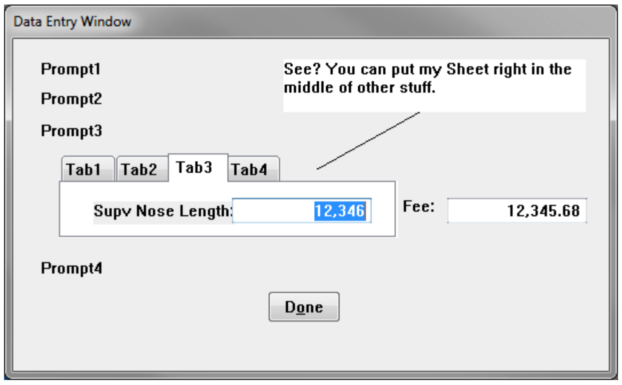


Figure 6. Tab 3 Selected

At run time, I can hide the tabs and the outline of the sheet:

```
?SHEET1{PROP:Wizard} = TRUE
?SHEET1{PROP:NoSheet} = True
```

So the running window looks good:

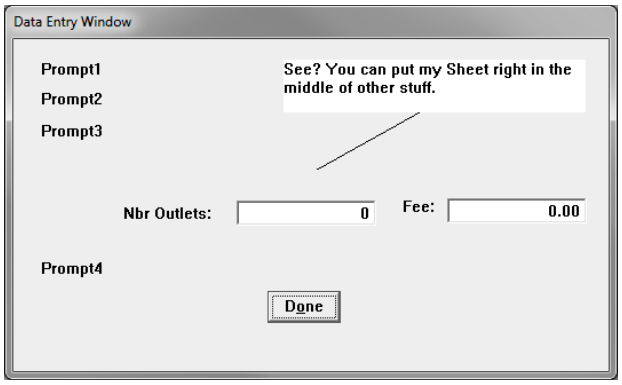


Figure 7. Running Window

(Clearly, I spent very little time aligning the prompts and fields on this window in the demo app. That's something that can safely be left until the app is debugged and working as expected.)

All that remains is ensuring that the correct tab is selected when the window opens. In `INIT, After Open the Window`, this code:

```
If Choice(?SHEET1) <> pMethod
  Select(?SHEET1,pMethod)
End
```

does the job, assuming I have previously read the configuration file and have a good value for my Method variable.

Creeping Elegance

Because I am working without a dictionary in the demo app, all the entry variables are locals.

In a real app, I have to talk to real file variables.

In fact, the Dictionary variable should probably be a Decimal, large enough on both sides of the decimal point to handle the largest possible value a user might enter. (The prototype for the calculating procedure would have be changed to (Long, *Decimal, *Decimal) but that's no big problem.)

At run time, I could pick up a flag from the configuration file and set the screen entry picture:

```
?MyField{Prop:Text}  = '@n05'
```

or

```
?MyField{Prop:Text}  = '@n8.3'
```

or I could continue using local variables on the tabs and assign them to the file variable in `TakeCompleted`.

At this point, too many choices.

Summary

The properties of Sheets are not well known and often overlooked. `Prop:Wizard` removes the tabs from a `Sheet`. `Prop:NoSheet` removes the lines that would otherwise outline the `Sheet`.

Remembering these properties makes customizing apps – without necessitating major re-writes or compromises – so much easier. Doesn't it?

Download the source

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

Article comments

BACK TO TOP

CLARION MAGAZINE
READ. LEARN. SOLVE.

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

A Windows 7 Alt-Lockup Fix For Clarion 6, Part 3

By Carl Barnes

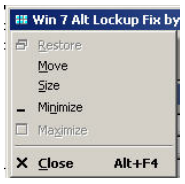
Posted October 29 2011

Reader feedback from Part 1 and Part 2 in this series prompted me to write this third part to address an additional problem keystroke. Readers also requested I create a template to make it very simple to fix an APP without needing to embed code. And I'll toss in an example of another windows message you can catch in the subclass procedure that will inform you when the user changes the system clock.

Alt+Space will lockup too

Mark Goldberg reported that pressing Alt+Space would also lockup the Clarion frame. I tested on a Windows 7 system with the Alt key lockup problem and the Alt+Space key locked it up too, darn! The purpose of that key is to drop the System Command menu (shown below) that provides functions for Move, Minimize, Maximize, Restore and Close. A trick I like is using that Move item to move a window that is entirely off the monitor back into view. Press the keys Alt+space, M, Up arrow and the window is in "move mode" and will move with the mouse.

The first step in the trick is to drop the System Menu and that requires the keyboard.



Fixing the subclass procedure code to spot Alt+Space is really simple. The menu key sent with the message has a value of decimal 32, the ASCII code for a space. I decided to have Alt+Space work the same as the Alt and F10 keys and drop the File menu. The other choice would be to have it do nothing, but that makes it less obvious how to get to the System Menu via the keyboard. My way the file menu drops, then pressing the left arrow will move to the System Menu.

Below is the original subclass code to spot Alt or F10:

```
CASE _wMsg
OF WM_SYSCOMMAND
   IF BAND(_wParam,0FFF0h)=SC_KEYMENU |
   AND _LParam=0 THEN
       !Send Alt+F message to frame to drop file menu
       PostMessage(_hWnd, WM_SYSCOMMAND, SC_KEYMENU, Alt1stMenu)
```

```
            RETURN(True)
         END
      END
```

The second line of the `IF` in the above code needs to be revised to spot Alt+Space key (=32) which is sent in the `LParam`:

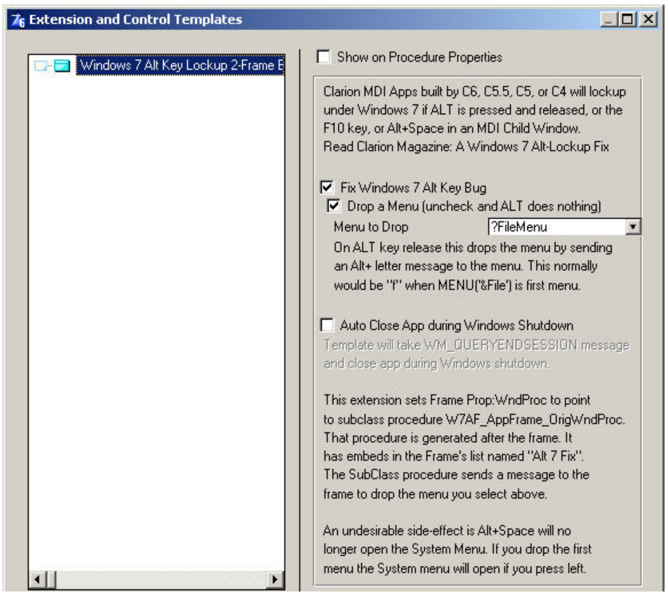
```
 IF BAND(_wParam,0FFF0h)=SC_KEYMENU |
      AND (_LParam=0 OR _LParam=32) THEN
```

The download included with the article has this new code. If you have already implemented the fix you can just change `_LParam=0` to `LParam=0 OR _LParam=32)`.

The second time it's a template

I'm pretty sure it was Don Childers who coined the phrase "The second time it's a template". It's a philosophy I agree with but don't always execute. I spent some time writing detailed instructions on how and where to embed the several code snippets. I tried to document some possible mistakes, errors and conflicts that could occur, and how to fix them. In less time I could have taken all of those requirements and coded the template to handle them. In a nutshell it's about `#PROMPT`ing the developer for anything optional and picking the right `#EMBED` for each `#AT`. Another big advantage of the template is when the requirements change (e.g. the additional need to handle Alt+Space), or you find a bug in the code, the generated code can be fixed in just one place, the template. The APPs just need to be recompiled.

The template file (CBAltWin7Fix.TPL) is included in the downloadable zip file. Put the TPL file in your C6 3rdParty\Template folder and register it. There is a single procedure extension template named `AltWin7Fix_Frame2` you must add to the Frame under the Extensions button. The template group is named `CBAltFix`. Below is a screen capture of the template properties page:


Extension and Control Templates
Windows 7 Alt Key Lockup 2-Frame E

Show on Procedure Properties [ ]

Clarion MDI Apps built by C6, C5.5, C5, or C4 will lockup under Windows 7 if ALT is pressed and released, or the F10 key, or Alt+Space in an MDI Child Window.
Read Clarion Magazine: A Windows 7 Alt-Lockup Fix

Fix Windows 7 Alt Key Bug [✓]
Drop a Menu (uncheck and ALT does nothing) [✓]
Menu to Drop: ?FileMenu
On ALT key release this drops the menu by sending an Alt+ letter message to the menu. This normally would be "f" when MENU('&File') is first menu.

Auto Close App during Windows Shutdown [ ]
Template will take WM_QUERYENDSESSION message and close app during Windows shutdown.

This extension sets Frame Prop:WndProc to point to subclass procedure W7AF_AppFrame_OrigWndProc. That procedure is generated after the frame. It has embeds in the Frame's list named "Alt 7 Fix". The SubClass procedure sends a message to the frame to drop the menu you select above.

An undesirable side-effect is Alt+Space will no longer open the System Menu. If you drop the first menu the System menu will open if you press left.

The template asks these questions:

Fix Windows 7 Alt Key Bug – Check this box and the Alt, F10 and Alt+Space keys will be processed by the subclass procedure. If you are using the template in C8 you can uncheck this because the C8 RTL does not have the bug.

Drop a menu – Check the box and the template will drop the selected menu when any of the three problem keys are pressed. If you don't like that behavior uncheck the box and those three keys will do nothing.

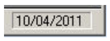
Menu to Drop - The drop list is populated from the top level menus. Normally you would drop the first menu, which is normally Menu('&File'). You pick the menu and the template finds the Alt+Key by searching for the "&". If the key changes the template adapts. This where the template improves on the hand coded solution as it always gets the right key.

Auto close App during Windows Shutdown – check the box and when the user shuts down windows your frame will not prevent it, it will get an `Event:CloseDown`.

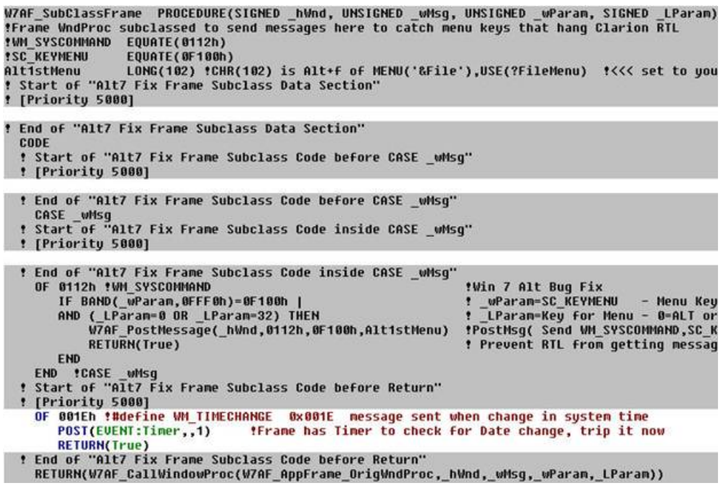
Handle time change message in template embeds

The template implements the subclass procedure in the same module as the frame. The subclass procedure does not appear in the procedure tree so I placed four embed points to allow inserting data and code to handle other windows messages sent to the frame. Because the template is attached as a frame extension those subclass embed points show in the frame's list of embeds, but they are not in the same scope as the frame. You cannot access any frame procedure data, just module data.

An example of a way the subclass embeds can be used involves the way my frame shows today's date in the status bar to remind users of the current date during data entry. Here's what that status bar segment looks like:



My frame has a timer that wakes up every 12 minutes, if the date changed it updates the status bar date. I figure nobody will be doing data entry at midnight so there is no need to update the date very often. What can happen is the system date gets changed by the users using the Windows utility, and when they don't see it immediately change in the status bar it confuses them. There's an easy fix for this problem. When the user changes the system clock Windows broadcasts a `WM_TIMECHANGE` message. Below is a screen capture of the entire subclass procedure including the embed code to process the message:



```
W7AF_SubClassFrame  PROCEDURE(SIGNED _hWnd, UNSIGNED _wMsg, UNSIGNED _wParam, SIGNED _LParam)
!Frame WndProc subclassed to send messages here to catch menu keys that hang Clarion RTL
!WM_SYSCOMMAND  EQUATE(0112h)
!SC_KEYMENU     EQUATE(0F100h)
Alt1stMenu      LONG(102) !CHR(102) is Alt+f of MENU('&File'),USE(?FileMenu)  !<<< set to you
! Start of "Alt7 Fix Frame Subclass Data Section"
! [Priority 5000]

! End of "Alt7 Fix Frame Subclass Data Section"
  CODE
  ! Start of "Alt7 Fix Frame Subclass Code before CASE _wMsg"
  ! [Priority 5000]

  ! End of "Alt7 Fix Frame Subclass Code before CASE _wMsg"
    CASE _wMsg
  ! Start of "Alt7 Fix Frame Subclass Code inside CASE _wMsg"
  ! [Priority 5000]

  ! End of "Alt7 Fix Frame Subclass Code inside CASE _wMsg"
    OF 0112h !WM_SYSCOMMAND                                !Win 7 Alt Bug Fix
       IF BAND(_wParam,0FFF0h)=0F100h |                    ! _wParam=SC_KEYMENU   - Menu Key
       AND (_LParam=0 OR _LParam=32) THEN                  ! _LParam=Key for Menu - 0=ALT or
           W7AF_PostMessage(_hWnd,0112h,0F100h,Alt1stMenu) !PostMsg( Send WM_SYSCOMMAND,SC_K
           RETURN(True)                                    ! Prevent RTL from getting messag
       END
    END  !CASE _wMsg
  ! Start of "Alt7 Fix Frame Subclass Code before Return"
  ! [Priority 5000]
    OF 001Eh !#define WM_TIMECHANGE  0x001E  message sent when change in system time
       POST(EVENT:Timer,,1)     !Frame has Timer to check for Date change, trip it now
       RETURN(True)
  ! End of "Alt7 Fix Frame Subclass Code before Return"
    RETURN(W7AF_CallWindowProc(W7AF_AppFrame_OrigWndProc,_hWnd,_wMsg,_wParam,_LParam))
```

The embed code spots the time change message (`OF 001Eh`). The frame already has timer embed code that checks the date, so posting an `EVENT:Timer` to thread one will run that existing code. The `Return (True)` it not really needed since it would be ok for the RTL to get this message.

You may be wondering why I did not use the data embed point to define an equate for `WM_TimeChange`. It's just my thinking that I need this equate for one line of code, so why slow down the compiler a few nanoseconds to store that symbol, then lookup once a few lines later. I always include a comment next to the value literal with the API define name so I can search for it later.

The Time change example above shows the basic method of dealing with subclass messages. If you need to communicate with the frame you'll need to `POST` an event to thread one. You can also use `NOTIFY` and it allows passing a long integer.

The first time it's a class

While templates are a powerful feature in Clarion they are not always the best solution. They are hard to write. They use a separate template language and you have to understand the application embed point model. The code you generate has to live with all other generated code and not cause conflicts. All the data and declarations must live in the same scope as all other APP code. That's part of the reason my template does not declare any equates, because they could conflict with other code.

There's another approach: the first time it's a class. When I asked my Realtor the three best things about OOP he said location, location and location. Actually he's right; the best thing about OOP is encapsulation. The location of the data, code and external declarations are all "encapsulated" in the class CLW file, with their own limited scope. The class is written using the same Clarion code you are used to writing. You just have to learn the Clarion OOP syntax and a few concepts. If you design it as a class from the beginning you encapsulate all the gory details and make it easy to plug into the APP with a few lines of embed code. Those few lines of code could be a template. I'll leave the class version for another day.

Summary

I hadn't been implementing the Alt fix for internal Apps figuring in house users could be trained to press Alt+Esc to recover. In reality it only takes 10 to 15 minutes to hand code the fix. My template reduces that to two minutes, and will do it perfectly the first time, every time. This will make the internal users very happy to not have (what appear to be) random lockups. My new goal is to focus on making the users happy, because I've recently realized I'm not just in the software business, I'm in the "people business". I learned that at Bruce Johnson's excellent DevCon 2011 talk "The Road Not Taken: What Robert Frost can tell us about programming in the 21st century. " I suggest you Google it and read the poem, it's very short. Then read an analysis of it because it is not about taking "the road less travelled".

Download the source

Carl Barnes is an independent consultant working in the Chicago area. He has been using Clarion since 1990, is a member of Team TopSpeed and a TopSpeed Certified Support Professional. He is the author of a number of Clarion utilities including CW Assistant, The CHM help class CHM4Clarion, and Clarion Source Search.

Article comments

BACK TO TOP

CLARION MAGAZINE
READ. LEARN. SOLVE.
Home Subscribe E-Books News Blog Store My ClarionMag My Lists Contact

Rationalizing Parameter Lists With Groups

By Steven Parker

Posted October 30 2011

In “Time and Time Again, Again,” I note that I had a fairly long parameter list in a class method. “Long” meaning hard to remember and, therefore, error prone when trying to use. The method prompting that observation is designed to split a DECIMAL variable into time units (days, hours, minutes and seconds, similar to how _FNSplit- breaks a fully qualified DOS file specification into its components).

The prototype for the procedure looks like this:

```
TimeSplit      PROCEDURE(*Decimal pTime,|
*Long pDays,   |
*Long pHours,  |
*Long pMinutes,|
*Long pSeconds)
```

In this procedure, only the first parameter, pTime, is passed into the procedure. The remaining parameters are values I want to get back (see “Returning Multiple Values”).

This prototype isn't anywhere near as unwieldy as one I created to set up complex filters using a series of SetFilter statements (see “SetFilter to the Max” and “Reports: OOP, ABC and Ignoring Templates, Part 2”):

```
SortRoutine    PROCEDURE(String pCalledBy,|
                    *Cstring pVNDFilter,     |
                    *Cstring pDeptFilter,    |
                    *Cstring pUSRFilter,     |
                    *Cstring pPLUFilter,     |
                    *Cstring pPartFilter,    |
                    *Cstring pSortParms,     |
                    Byte <pLowVendor>,       |
                    String <pVNDList>,       |
                    Long <pStoreID>),Byte
```

Here, the first and last three parameters contain information. The remaining six parameters are intended for return to the the calling procedure.

(Please picture each of these without the continuation characters. Without the continuation characters, these prototypes are almost unreadable.)

The only solace I take in these prototypes is that, as long and messy as they are – and they are – they pale in comparison to what I saw in Charles Petzold's *Programming Windows* (that's Windows 3.0, 1998) in a “Hello World” example. But, it is a small solace.

Procedure prototypes like these cause two concerns. First and most obvious is that typing in these long prototypes, in the limited space available – less limited in C7/8 than previous versions of Clarion, to be sure, but the entry area is still small – creates numerous “typotunities” (a neologism meaning “opportunities for typographical errors”). F10 functionality, making a larger text box available for entry, is not available on the Prototype line.

The second concern is that calling the procedure, getting the parameter list correct is also a major typotunity. I either have to remember the parameter list (right!) or I have to make paper notes to refer to while coding (with my handwriting … right!).

The calls for the two procedures referenced look like this (continuation characters added):

```
MyStarDate.TimeSplit(GrossDifference,       |
DayComponent,       |
HourComponent,      |
MinuteComponent,    |
SecondsComponent)
```

and

```
If SortRoutine(pCalledBy,     |
VNDFilter,     |
DeptFilter,    |
USRFilter,     |
PLUFilter,     |
PartFilter,    |
SortParms,     |
LowVendor,     |
VNDList)
```

The answer, of course, is `Group` parameters. Tony York, in “Clarion 101: Passing Groups,” aptly observes “Groups provide a nice neat way of passing across a 'single' record of information, coupled with a nice simple prototype for the receiving procedures.” The answer to all my problems!

Not only does a `Group` structure allow referring to a substantial quantity of data in many fewer keystrokes, a local group also makes a report on form data, without having to save the form, easy. In fact, running a report before saving the form was Tony's motivator for the article.

Tony's need is “one way.” He needs to pass a large number of fields from one procedure to another but avoid the typotunities I mentioned. My need is “two way.” I need to get values back from the called form.

Will `Group` parameters do for me?

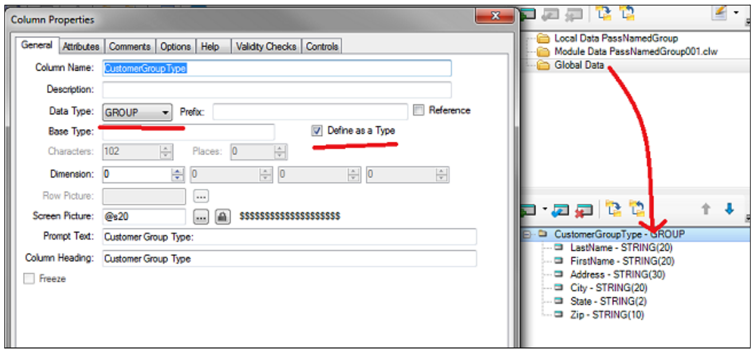
Group Parameters

The documentation on passing Named Groups, the proper nomenclature for passing `Group`s as parameters, is sparse indeed. I recall seeing it at one time but now can only find “Named Group” referred to under "Reference Variables" (and I have a bit of a problem with Reference Variables). So, using Tony's article as my guide, I decided to try passing the `Group` by address. (Passing by address is the standard method of ensuring that both the calling and called procedures operate on the same variables, not on copies of the variables.)

It's just a few simple (without Tony's article, I would never have anticipate this later adjective, given some of the examples I've seen in others' apps) steps.

Step 1: Create the prototype declaration for the `Group`

The first thing to do is create the prototype, the `TYPE`d declaration, of the `Group`. This can be done either in a global data embed or in the (global) data pad. I chose the data pad, as it writes large parts of the structure for me:



Column Properties

General | Attributes | Comments | Options | Help | Validity Checks | Controls

Column Name: CustomerGroupType
Description:
Data Type: GROUP
Prefix:
[ ] Reference
Base Type:
[✓] Define as a Type
Characters: 102
Places: 0
Dimension: 0 0 0 0
Row Picture:
Screen Picture: @s20 $$$$$$$$$$$$$$$$$$$$
Prompt Text: Customer Group Type:
Column Heading: Customer Group Type
[ ] Freeze

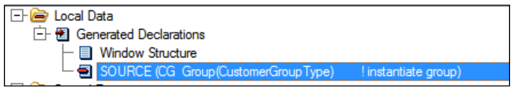
```
Local Data PassNamedGroup
Module Data PassNamedGroup001.clw
Global Data

CustomerGroupType - GROUP
    LastName - STRING(20)
    FirstName - STRING(20)
    Address - STRING(30)
    City - STRING(20)
    State - STRING(2)
    Zip - STRING(10)
```

Figure 1: TYPEd Group Declaration

Using a data pad also exposes the variable declarations to the Window Formatter. As an inveterate "Designer user," this is important to me (not for the TYPEd declaration, of course, but as a general "style" choice).

Step 2: Create an actual Group

A `TYPE`d data declaration allocates no memory. That is, it does not actually declare any data. It is just a prototype on which real data is modeled. So the next step is to create an actual `Group`. This time I used a data embed, in the calling procedure:



```
Local Data
    Generated Declarations
        Window Structure
        SOURCE (CG  Group(CustomerGroupType)        ! instantiate group)
```

Figure 2: Creating a Real Group

The embed contains this code:

```
 CG   Group(CustomerGroupType)        ! instantiate the group
      END
```

Strictly speaking, this step is not always necessary. See "Tech Notes," below.

Step 3: Use the `Group`

This is the important part.

I created a `Window` procedure and some local data declarations. I populated the local data on the window:

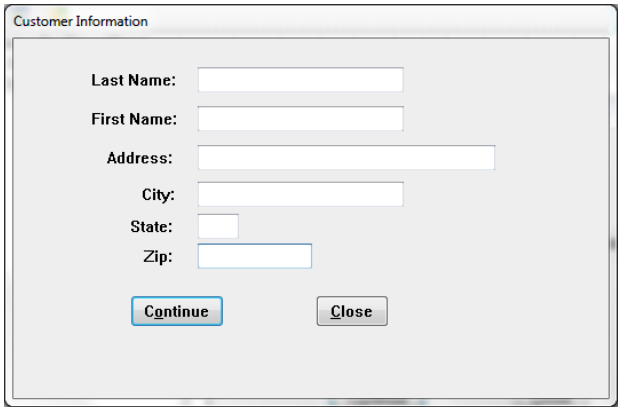


Figure 3: Data Collection Window

When the user presses the “Okay” button on this window, I load the instantiated Group from the window variables (N.B.: when I created the local group for the display variables, I didn't use a prefix; so, all of the code uses dot syntax – also note that the duplicate declaration errors Tony mentioned in his article may show up):

```
CG.LastName = LOC:CustomerGroup.LastName ! copy screen vars to group
CG.FirstName = LOC:CustomerGroup.FirstName
CG.Address = LOC:CustomerGroup.Address
CG.City = LOC:CustomerGroup.City
CG.State = LOC:CustomerGroup.State
CG.Zip = LOC:CustomerGroup.Zip
```

and I call the next procedure with the Group as its parameter:

```
CalledProc(CG)                    ! pass group to called proc
```

Strictly speaking, this syntax is not always optimal. Because, in this case, I created a local GROUP, with the same structure as my TYPEd GROUP, I could have passed my local GROUP directly:

```
CalledProc(LOC:CustomerGroup)
```

See “Tech Notes,” below.

On return from the called window, I re–load the window variables from the `Group` (just in case they're different – got to see the changes before I can use them):

```
LOC:CustomerGroup.LastName = CG:LastName    ! reassign "returned"
                                            ! values to screen vars
LOC:CustomerGroup.FirstName = CG:FirstName
! assign the rest of the display variables
```

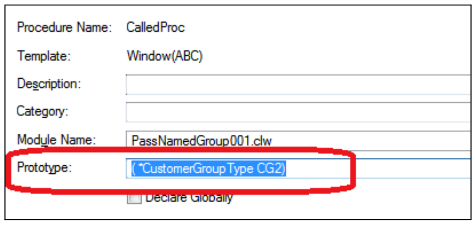
and force the window to redisplay the data:

```
ThisWindow.Reset(1)                                  ! refresh window
```

If the data is changed in the called procedure and it does not appear after the refresh, I know I have … gone astray.

As I shall discover, later, re–assigning and resetting is not necessary.

Step 4: Prototype the called procedure



Procedure Name:	CalledProc
Template:	Window(ABC)
Description:	
Category:	
Module Name:	PassNamedGroup001.clw
Prototype:	(*CustomerGroupType CG2)
	Declare Globally

Figure 4: Prototyping the Called Procedure

This is where the work is done.

The prototype is:

```
(*CustomerGroupType CG2)
```

`CustomerGroupType` is the “Named Group” created in Step 1 (spell it correctly! C8 is not as forgiving as earlier versions which, I found, would allow typos in the group's label). `CG2` is the local label for the data.

In `INIT`, I assign from the “incoming” Group to local screen variables. And, when the user presses “Okay,” I re–assign from the screen variables back to the Group parameter for the return trip. Note that there is no instantiation of a Group in this procedure. (Read that again; it's significant.)

I could *have skipped a local declaration and this local assignment. I* could *have used the Group variables directly. But populating variables that are not on a data pad is not as easy as using variables that are on a data pad. I chose to this in order to use the Window Formatter; if you are comfortable populating variables with minimal IDE assistance (and, using dot syntax), the local declarations (and, therefore, the re–assignment) are entirely unnecessary.*

Oh, yes, I also close the window.

But! Does It Work?

There is a demonstration app, downloadable at the end of this article (there is a second app, by Dave Harms, with all the “strictly speaking” stuff removed – much cleaner). They are both made with Clarion 8.0.8658 and I have included compiled EXEs and all the run times you need to run them.

Here's what you should do. Start the app. Enter some information on the first window.

Press “Ok.” If you see the information on the second window (I placed different captions on each window so you can tell which window you're on), the “pass” part worked.

Press “Ok” to exit the second window. Press “Close” to exit the app.

Now, start the app again. But, instead of completing a name and address on the first window, just press “Ok.”

On the second window, which should be empty, enter a name and address. Press “Ok.”

If the information appears on the initial window, when you return to it (check window captions to make sure you're in the right places!), the “return” part worked ….

If you have Clarion 7 or 8, try removing the asterisk from the prototype of the called procedure. You should notice no difference in the app's behavior. It seems that Group parameters are passed only *by address.*

Tech Notes

The important part, the critical part, of passing named `GROUP`s is the `TYPE`d `GROUP` declaration. This declaration creates a new data type; the label of the `GROUP` functions just like "`STRING`," "`LONG`" or "`REAL`." Thus, when I use that label, Clarion "knows" what the data looks like and what to expect. If the data passed matches, *exactly*, the compiler is happy. On the other hand, if the data being passed is so much as one byte different, the compiler will throw an error.

What matters is that the data match the declaration (in the `TYPE`d `GROUP`). What does not matter is the label of the data in the caller. The label in the called procedure doesn't matter either; the structure of that datum is fully determined by the `TYPE`d `GROUP` declaration.

In the first demo app, I made a local `GROUP` *exactly* matching the `TYPE`d declaration. Because I declared a structure precisely matching what Clarion expects, I didn't really need the local instantiation of the `TYPE`d declaration.

Neither did I need the assignments to and from the local variables. The procedure call becomes much simpler:

```
CalledProc(LOC:CustomerGroup)
```

On the other hand, if I was passing a subset of a file's fields or a set of temporary variables (as is the case in the two prototypes at the top of this article), I *would* need a (local) `GROUP`, to ensure the compiler stays happy. I would still *not* need the local instantiation of the `TYPE`d declaration.

Similarly, because `GROUP`s are inherently passed by address, re-assignment on return it unnecessary.

Coming Soon

Can I pass a `Group` back and forth across a thread?

Download the source

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

Article comments

by Mark Riffey on November 5 2011 *(comment link)*

One thing worth noting is that the use of typed groups like these also make your code far more sturdy from a maintenance perspective.

BACK TO TOP

Passing Named Groups Across Threads

By Steven Parker

Posted October 31 2011

In my last article, I showed an implementation of passing Named Groups. My intention was to pass a large quantity of data, such that a prototype enumerating each variable would be truly unwieldy (or, at least, a real PITA), to another procedure *and* to get back any changes made to that data.

Now it is time to see if I can pass a Named Group to a `START`ed procedure. And I still want to get back any changes. (Do I *really* need to pass Named Groups across threads? No more than I need to START an update procedure. In other words, not at all. But I'm equally certain that someone out there in Clarion Magazine reader-land *will* want to do this. In fact, a few weeks after a draft of this article was submitted, a long thread was started in one of the news groups on just this topic. So as long as I'm on a roll, I may as well document "passing groups across the thread.")

In the last article, I called the second procedure as follows:

```
CalledProc(LOC:CustomerGroup)
```

where LOC:CustomerGroup is a local structure *precisely* matching a global `TYPE`d `Group`.

The question before me know is, "Can I START CalledProc?"

Of course,

```
Start(CalledProc(LOC:CustomerGroup))
```

is syntactically invalid and, therefore, will not compile. Checking the LRM shows that the `START` call, when there are parameters, must read:

```
Start(CalledProc, ,LOC:CustomerGroup)
```

Of course, this will not work either. It won't even compile. Why? `START`ed procedures can only take `String` parameters. A `Group` is not a `String`, the compiler knows this, so it won't work.

Since it can't be done, I'm going to show you how to do it.

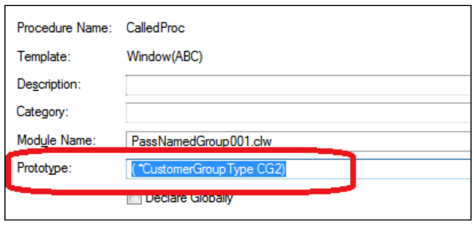
Of Groups and Strings

Consulting the LRM again, the spec for "`Group`" states:

When referenced in a statement or expression, a `GROUP` is treated as a `STRING` composed of all the variables within the structure.

In other words, with a bit of prudence, a `Group` and a `String` are (or can be) interchangeable.

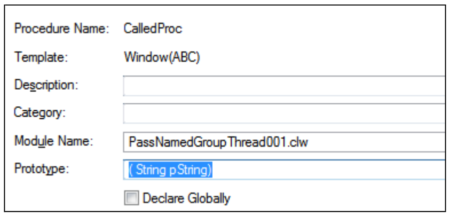
The prototype I used, in the last article, for the called procedure:



Procedure Name: CalledProc
Template: Window(ABC)
Description:
Category:
Module Name: PassNamedGroup001.clw
Prototype: (*CustomerGroupType CG2)
Declare Globally

Figure 1: Previous Prototype

can be changed to receive a `String`:



Procedure Name: CalledProc
Template: Window(ABC)
Description:
Category:
Module Name: PassNamedGroupThread001.clw
Prototype: (String pString)
Declare Globally

Figure 2: New Prototype

In essence, I'm letting the compiler *cast* my `Group` to a `String`.

The application will now compile. It will not run correctly – the first window works fine, the second receives no data - but it does compile. That's progress (of a sort).

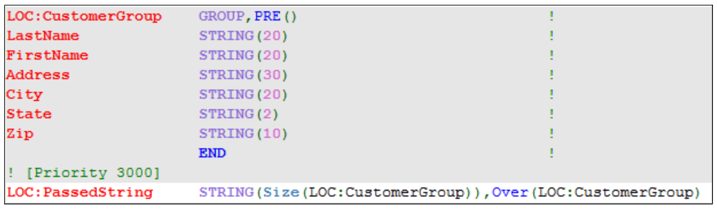
Of Strings and Groups

The problem, of course, is that the procedure receives a String. But the window, as I designed it, displays local variables. These local variable happen to be part of a local `Group` (as mentioned in the previous article, I did this so that I could use the Window Formatter with minimal effort on my part, while having a structure of the precisely correct specification). Obviously, I need to transfer the incoming data from the `String` to the `Group` and to its constituent fields.

Starting with the .APP from the last article (the Harms Variation), I created a local string, `LOC:PassedString`, to "receive" the incoming parameter. My thinking is to place it `OVER` the local `Group` used for display/entry on the window. (And that is why I can't use the passed datum directly, as you will soon see.) Initially, I made the string much larger than I knew was needed.

The compiler doesn't much care for an `OVER`ed variable being larger than variable it is `OVER`ing (in the past, the compiler only complained when the `OVER`ed variable was smaller than what it was placed `OVER`; note this change). I didn't think hard coding the size of the `Group` was a good idea (just in case I ever decide to put more fields in the `Group`). What to do?

I decided that if the compiler is unhappy with what I'm doing, then the compiler should handle the size issue. And, it can, thanks to the `SIZE` directive:



```
LOC:CustomerGroup     GROUP,PRE()                                  !
LastName              STRING(20)                                   !
FirstName             STRING(20)                                   !
Address               STRING(30)                                   !
City                  STRING(20)                                   !
State                 STRING(2)                                    !
Zip                   STRING(10)                                   !
                      END                                          !
! [Priority 3000]
LOC:PassedString      STRING(Size(LOC:CustomerGroup)),Over(LOC:CustomerGroup)
```

Figure 3: How to make the compiler happy, and do the work for me

With the compiler now satisfied, in `INIT`, I can get the passed parameter into my local variable with a simple assignment:

```
LOC:PassedString = pString      ! copy parameter to local vars
```

Since `LOC:PassedString` is `OVER` the `Group` used for the screen fields, I have effectively assigned the incoming datum to that `Group` also. "Loverly," as Eliza Doolittle might say.

This is *all* that is required to move a `String` parameter into the local variables and display them correctly on the window. As mentioned, it also guarantees that I have the requisite `Group` in the called procedure.

If you open the demo app (you can download it at the end of this article; it is made in Clarion 8.0.8658 but I have included a compiled EXE and all the run times needed) and enter a name and address on the first screen, you will see that the information you entered appears on the second screen when you press "Continue." So passing a `Group` across a thread to a `START`ed procedure does work.

Returning to the Caller

The problem with getting back any changes is that I `START` the called procedure. Once the second window opens and hits its `ACCEPT` loop, `ACCEPT` in the calling procedure also starts executing again. So, the normal tactic of re-assigning variables to the passed `STRING` just before returning to the caller won't work.

This code:

```
pString = LOC:PassedString   ! re-assign screen vars back to
                             !  passed string var
```

won't work because the caller is no longer "listening" to the second window. `Accept` has done its Elvis imitation and left the building.

Another, somewhat more powerful, reason I have trouble retrieving any changes made in the second window is that any changes made to the variables in the second window occur after the first window has `RESUME`d. In other words, whatever the contents of the `Group` when the first window's `ACCEPT` loop resumes is all that the first window knows about (the variables in the second window are a copy of the `Group` to a local string...). Again, all changes made on the second window occur after `ACCEPT` resumes … well, you see the problem.

Talking to Another Thread

Since the calling procedure/thread has moved on, I obviously need a way to tell that procedure/thread that I am done with the called procedure.

There aren't very many ways to effect inter-thread communication. And, most of them expect me to know the number of the thread I want to communicate with. In "Windows-speak" this is "sending a message to another thread, in this case the launching thread (why the app can't figure it out for itself just baffles me … I blame Windows®).

To make this possible, I declare a global (and, therefore, static) variable to hold the caller's thread number:

```
MainThread Long,Auto
```

and early in `INIT` of the calling (first) procedure ensure that it has a usable value:

```
MainThread = Thread()   ! for inter-thread communication, later
```

(Dave Harms notes that I could avoid a `Global` variable by passing `THREAD()` – as a string – to the called procedure. How? `START` can take up to three parameters and I'm only using one.)

I can think of three ways – though I'm sure there are more – of communicating from the called procedure to the calling procedure:

- Create a global `STATIC` variable and, in the called procedure give it a value just before returning; use `EVENT:TIMER` in the caller to check (and clear, if necessary) this variable
- Create a user defined `EVENT` and `POST` that event to the calling thread
- `NOTIFY` the caller

There may also be Windows APIs suitable to this purpose. But I don't happen to know them (if you do or if you have better ways to effect inter-thread communication, a follow on article would be much appreciated, please).

I won't argue that the first mentioned methods is … well, it's stupid.

`POST` has some potential. `POST` lets me specify the thread I want to communicate with:

```
POST( event [,control] [,thread] [,position] )
```

I do have to define my own “event” (see “User Defined Events” in the on line help) to avoid triggering some standard event. So, for example:

```
Post(Event:HeyDudeTheGroupIsBack, , MainThread)
```

would ensure that the calling thread knows I am done on the second window. (Assuming, of course, that I previously set up an `EQUATE` for `Event:HeyDudeTheGroupIsBack`, preferably with a value like `Event:User+`*`somenumber`*.)

But.

But, the contents of the `Group` are current only on the called thread. The window data is invisible to the calling thread (by definition). How can I `POST` the data in the calling thread?

So far as I can tell, I can't. `POST` can tell the calling thread something has happened, even what happened (because I defined the “what”). But I don't see how to get the data from the called to the calling thread.

I have never worked with `NOTIFY` (and it's partner `NOTIFICATION`) before. But the LRM description of `NOTIFY` looks promising:

```
NOTIFY( notifycode, <thread>,  <parameter> )
```

Better still, every Window template-based procedure has a built in embed, `EVENT:Notify`, for trapping notifications sent to it:

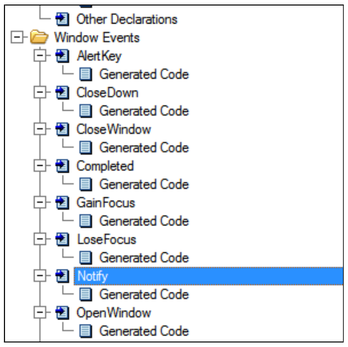


Figure 4: Built in EVENT:Notify

The LRM states this about `Notify`:

> **NOTIFY** Sends an event and optional parameter to a receiving thread. *notifycode* An unsigned integer value that indicates the notification or request code. *thread* A signed integer that identifies the number of the receiver thread; if omitted or equal to 0, the current thread is the receiver. *parameter* An optional LONG value that is used as a parameter of the notification or request

"*notifycode*" looks a lot like `POST`'s "event" parameter, except that it doesn't require me to make a declaration similar to a User Defined Event (though I can and probably should). The "*thread*" parameter, well, I understand that well enough also.

The "*parameter*" parameter, on its face, looks like a datum I can "send" to "*thread*." Except that it is a `LONG` and I need a `Group` or a `String`.

But!

It occurs to me that once I re-assign the local variable to the passed `*String` in the called procedure:

```
pString =  LOC:PassedString   ! re-assign screen  vars to passed var
```

(or, I could avoid the assignment to the passed string by passing back the address of the `Group` I'm already using – but I didn't see this until Dave Harms pointed it ot to me) I *do* have a `LONG` that I can use in `Notify`. I have the address of `pString`:

```
Notify(9999,MainThread,Address(pString))
```

or

```
Notify(9999,MainThread, Address(LOC:PassedString))
```

("9999" is my *notifycode* – just for this sample app, I did "a lazy" instead of declaring an `EQUATE`. Note: there is a second demo app featuring passing `LOC:PassedString` back directly, without an intermediate assignment.)

`Notification` is how I retrieve the "*parameter*" sent by `Notify`:

```
NOTIFICATION(notifycode,  <thread>, <parameter> )
```

I'm working on the presumption that the data at `Address(pString)` will be available back in the caller. I am assuming that the memory will not have been overwritten yet because the variable represented by `pString` is actually owned by the calling procedure, which is still in scope.

Using `Notification`, however, is a bit of an adventure. While `Notification` is declared in Builtins.CLW, I found that I *had* to declare two variables, for the first and third parameters:

```
nCode     UNSIGNED,Auto
nlParam     Long,Auto
```

Without my own declarations, the compiler refused to accept the prototype of the `Notify` call.

With these variables manually declared, I'm “good to go.” In `Event:Notify`, I can act on my `Notify` call (because of the *`notifycode`* parameter, I can have one or more `Notify's`):

```
IF NOTIFICATION(nCode,,nlParam)
     Case nCode
     Of 9999
       ! do stuff
```

Now all I need is a way to use `Address(pString)` to retrieve the actual data.

```
Peek(nlParam,LOC:CustomerGroup)
```

seems like a good candidate (again, `Peek` is something I've never used before).

The final code looks like this:

```
IF NOTIFICATION(nCode,,nlParam)
    Case nCode
    Of 9999
       Peek(nlParam,LOC:CustomerGroup)! picks up address of group and
                                     !  writes it to local vars
    End
  End
```

I’m told you can use the MemCpy API call in place of `Peek`, but as both do the same thing and `Peek` requires one less parameter and no call to `Address()` it seems like the better choice.

But does it work?

In the demo app, leave the first window empty and press “Continue.” On the second screen (I made sure the window captions will tell you which window you're working on), enter a name and address. If this works, when you press “Ok” on the called window, you should see your entry on the caller's window. Try it.

Summary

I can see a use for passing a large quantity of data via a `Group` to a `START`ed procedure. If I need to start a background computation at the end of a data entry window (completing a transaction in a POS or order entry app come immediately to mind), `START`ing the end of transaction processing off to another thread should allow the program to continue more quickly and more smoothly.

Why one might want to `START` another procedure *and* receive data back from the spawned thread … I don't understand why someone would want to do this. But, if ever I do need to, I know that I can.

And. At the recent CIDC Devcon, Mark Goldberg told me he's been passing Groups between threads for a long time. He uses reference variables and has agreed to do an article on that method of passing `Group`s between threads. I'm looking forward to reading it.

Download the source

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

Article comments

by surfersteve on November 1 2011 (comment link)

had you though of just using the &amp;= (addressof group), this even works across dynamically loaded dll's where all you pass is the address in a procedure from on dll to another.

it even works for interfaces that let you load a dll depending on the configuration you might adopt.

the interface can then be assigned locally and you dll can access anything you support in the other dll or exe.

a much under used method of creating more dynamic apps, similar in nature to a dot net app, you can do this for your standard clarion exe app, but of course ABC as never designed as an interface base development system to the level where it could have create a dot net system itself.

BACK TO TOP