Clarion Magazine

Clarion News

- » C7 IDE Tools Movie
- » J-Cal Adds Multi-Day Appointments
- » C7 IDE Movie
- » C7 Project Conversion Movie
- » Autodetect Menu Styles in C7
- » Post News At ClarionLife.NET
- » FileTuner Maintenance Codes Extended
- » BoxSoft Super Tagging 6.60
- » StrategyOnline Simplified Forms
- » J-Media 1.01
- » FileTuner 0.30
- » RPM eXtended Target Released
- » BoxSoft Super Stuff 6.60
- » XFiles 1.6
- » Clarion Third Party Profile Exchange Adds Web Update
- » StrategryOnline Releases J-Media
- » C7 Styled Menus Example App
- » C7 Project System Movie
- » C7 Visual Styles Movie
- » C7 On PimpMyClarion
- » J-Fax 1.60
- » J-Skype 1.50
- » Fomin Report Builder 3.10
- » Product Scope 7 Introductory Pricing
- » Clarion Desktop 4.0

Latest Subscriber Content

The New Dictionary Diagrammer

In a recent alpha build SoftVelocity included an early release of the new Dictionary Diagram tool. Dave Harms reports.

Posted Sunday, September 30, 2007

Clarion.NET Language Changes: What's New

Clarion.NET opens up a whole new world to Clarion developers, and the Clarion language is evolving to take advantage of that world. Dave Harms explores the new language features in Clarion.NET.

Posted Thursday, September 27, 2007

Clarion.NET Language Changes: What's Gone

The .NET platform is a lot richer than Win32, but it also imposes some requirements that aren't fully compatible with Clarion as we know it. In this first of two articles on Clarion.NET language changes Dave Harms looks at what's going away.

Posted Thursday, September 27, 2007

Debugging SQL Problems With File Callbacks

Sometimes database problems don't respond to the usual array of debugging tools. Marty Honea shows how to find out exactly what the driver is doing with file callbacks.

Posted Wednesday, September 19, 2007

Multiline Radio Buttons And Checkboxes

Like some other Win32 development tools, Clarion lacks support for multi-line radio buttons and checkboxes. But adding that support, via the Windows API, is a lot easier than you might think. Carl Barnes explains.

Posted Tuesday, September 18, 2007

Dylan Does ClarionMag

Check out this nifty video of Bob Dylan singing about Clarion Magazine.

Posted Monday, September 17, 2007

Waiting For Files With Clarion Threads

Alan Telford revisits Jim Kane's C5 example code for detecting when a file appears in a directory, and replaces Jim's OS thread creation with a simpler version using C6's preemptive threads.

Posted Wednesday, September 12, 2007

- » InBack 1.7
- » Clarion.NET Cool Buttons Video
- » Clarion 7 Beta Out To CSP Members
- » Oak Park Solutions Server Plans On Sale
- » AmazingGUI Glassy Buttons Project Preview
- » Clarion Desktop 4.0
- » CPCS Office Schedule
- » Icetips Magic Locks
- » UKCUG New URL
- » EasyOpenOffice 1.05

[More news]

[More Clarion 101]

Latest Free Content

- » Dylan Does ClarionMag
- » Getting back on schedule...
- » Source Code Library 2007.08.31 Available

[More free articles]

Clarion Sites

Clarion Blogs

The Clarion 7 Beta Release

Dave Harms unpacks the Clarion 7 beta release, surveys the available features, and explains the interim process for compiling C6 applications in C7.

Posted Wednesday, September 12, 2007

Getting back on schedule...

For more on what's coming this September read the latest ClarionMag blog entry.

Posted Friday, September 07, 2007

Source Code Library 2007.08.31 Available

The Clarion Magazine Source Code Library has been updated to include the July/August source. Source code subscribers can download the January-August 2007 update from the My ClarionMag page. If you're on Vista please run Lindersoft's Clarion detection patch first.

Posted Thursday, September 06, 2007

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

Source Code

The ClarionMag Source Code Library

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

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

More info • Subscribe now

Printed Books & E-Books

E-Books

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

Printed Books

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

- » Clarion 6 Tips & Techniques Volume 3 - ISBN: 0-9689553-9-8
- » Clarion 6 Tips & Techniques Volume 1 - ISBN: 0-9689553-8-X

- » Clarion 5.x Tips and Techniques, Volume 1 - ISBN: 0-9689553-5-5
- » Clarion 5.x Tips and Techniques, Volume 2 - ISBN: 0-9689553-6-3
- » Clarion Databases & SQL - ISBN: 0-9689553-3-9

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

From The Publisher

About Clarion Magazine

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

Subscriptions

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

Satisfaction Guaranteed

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

Dave Harms

ISSN

Clarion Magazine's ISSN

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

About ISSN

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

Clarion Magazine

Clarion News

Search the news archive

Chicago CUG Oct Meeting

The next meeting of the Chicago Clarion User Group will be Wednesday October 10th, 2007 at 6:30 p.m. It will be held in suite 270, in Nebo Systems offices. The address is 1 South 450 Summit Ave. Suite 270, Oakbrook Terrace, IL 60181. The sign at the front of the office complex says Summit Oaks. Steve Parker will be presenting Clarion 7.

Posted Wednesday, October 03, 2007

Tagkeys Uses Cool Frames

A new Tagkeys update is available that uses Purple Swift's Cool Frames.

Posted Monday, October 01, 2007

StrategyOnline Volume Discounts

StrategyOnline has implemented volume discounts on all its products.

Posted Monday, October 01, 2007

J-Media 1.02

J-Media 1.02 has been released and is available for download. This build includes the ability to record audio.

Posted Monday, October 01, 2007

awDebugger 1.1

awDebugger 1.1 has been released. The most important features added in this release are a new Watch Window for adding variables and expressions that will be evaluated at each step, and the ability to inspect variable values by just placing the mouse cursor over a variable name in the source code viewer (a tooltip appears displaying the variable name, data type & current value). You can also select any text in the source code viewer, and as long as the selected text is a valid Clarion expression the selected expression will be evaluated and its result will be displayed in a tooltip as well. Free trial available.

Posted Monday, October 01, 2007

1stLogoDesign Fall Sale

1stLogoDesign's Corporate Identity Package is now $200 OFF. Package includes: Corporate logo package: Stationery design package; Bronze web design; Free delivery inside the U.S. Ready-to-go logo packages are now 20% off.

Posted Monday, October 01, 2007

SetupBuilder 6.6 Build 2000

Lindersoft has released SetupBuilder 6.6 Build 2000. Lindersoft strongly recommends customers upgrade to the latest version of SetupBuilder 6.6 as soon as possible to maintain the highest level of support, performance and reliability. If you have a current SetupBuilder Maintenance and Support Subscription Plan, the update is free of charge. You can get the latest version by selecting "Check for Updates" from within the SetupBuilder 6 IDE.

Posted Monday, October 01, 2007

Search Engine Profile Exchange Updated

The Search Engine Profile Exchange includes: General Internet Interest; Software Authors; File Submission Tools; PAD Enabled sites and tools; File Search Engine.

Posted Monday, October 01, 2007

Evolution Browse Export Professional 1.12

New in Evolution Browse Export Professional 1.12: Includes the source code procedure from exportation screen; New procedure for saving inquiries; New procedure to load an inquiry; The exportation event works as a wizard.

Posted Monday, October 01, 2007

ClarioNET Payment Gateway

The payment gateway on the ClarioNET web site has gone live. Upgrades and new licences can now be purchased via credit card. The payment process itself is actually carried out through a ClarioNET application. When you press the BUY NOW button, you will be able to download the client side and start the whole process.

Posted Monday, October 01, 2007

Google Maps Example

Richard Rose has posted an example app of how to call Google Maps from within a Clarion app. Available in the Par2 download section.

Posted Monday, October 01, 2007

C7 IDE Tools Movie

Bob Foreman is one busy guy. He's posted yet another movie on C7, this one 14 minutes long. A nice touch this time is a downloadable zip in addition to the usual browser view. Features covered include: IDE languages; Ambiences (different appearances for different languages); Load/save options; The task list (which is configurable); Output window appearance; File locations; Code snippets; Text editor settings including XML editing; Customized highlighting; Multiple Clarion versions; Window and report designer options (C7 and Clarion.NET); Adding external tools to the IDE.

Posted Wednesday, September 19, 2007

J-Cal Adds Multi-Day Appointments

J-Cal now supports appointments that span multiple days.

Posted Wednesday, September 19, 2007

C7 IDE Movie

Bob Foreman narrates this seven minute movie on the new IDE's configurability. Features covered include the online help and tips on using the various pads (dockable, floatable, pinnable windows). Bob also goes into the various layouts and how you can customize your work environment.

Posted Wednesday, September 19, 2007

C7 Project Conversion Movie

Bob Foreman narrates this six minute movie on converting applications from C6 to C7, including using the conversion wizard to move multi-DLL applications into a single solution. Bob also covers some project option settings, including setting the build output detail level.

Posted Wednesday, September 19, 2007

Autodetect Menu Styles in C7

Bob Zaunere has posted a blog entry on new functionality for detecting the user's current menu styles/themes. An EXE demonstrating the technique is available; the new feature is implemented for the next beta build.

Posted Wednesday, September 19, 2007

Post News At ClarionLife.NET

If you want to post your own news items at clarionlife.net please send an email requesting author permission to zero AT-NO-SPAM clarionlife.net.

Posted Wednesday, September 19, 2007

FileTuner Maintenance Codes Extended

Development on FileTuner has resumed, and new maintenance plan codes have been issued to all original FileTuner purchasers (some of the original codes had expired).

Posted Wednesday, September 19, 2007

BoxSoft Super Tagging 6.60

BoxSoft Super Tagging 6.60 is now available. This release is compatible with the C7 beta.

Posted Wednesday, September 19, 2007

StrategyOnline Simplified Forms

StrategyOnline forums now have reduced number of categories per product.

Posted Wednesday, September 19, 2007

J-Media 1.01

J-Media 1.01 is available for download. Changes include: Complete shipping support for the video control; New demo; Usable in an app that runs off a CD-ROM; Video filters; Simplified class code; Fixed demo uninstall.

Posted Wednesday, September 19, 2007

FileTuner 0.30

New in FileTuner 0.30: Legacy version and example for C6.x; Automatically takes the file owner from the file attributes when possible. FileTuner is a Topspeed and Btrieve file tester, refresher and fixer. It is a template coupled with an app, which has a Browse procedure that combined with a template will take any DCT you specify, and will allow you to test, scan and fix damaged files. The procedures in the app can be imported into yours, or you can change the DCT of the app.

Posted Wednesday, September 19, 2007

RPM eXtended Target Released

A new release of RPM eXtended Target for Clarion 6.3, build 9051+ is now available. Although legacy files are included, please note that extended target support is currently only for ABC applications. Legacy support will follow in a few weeks. Updated installs for other Clarion 6 builds and C7 Beta will be released in coming days. This release provides support for current SoftVelocity report targets and most, if not all, third party extended report targets. There's enough flexibility built into this release that you can even create your own targets, if you have the need. In addition this release will allow you to intercept the action before and after target generation. A developer callback method is included so you can run any routine that's local to the report or any procedure that is available. This is accessible from the destination dialogs as well as the developer procedure. Page ranges, as has always been included in RPM, are supported for the extended targets. Also included is the option to restrict range support for individual targets. Programmatic forced target generation support is provided so that targets are processed immediately after the report has completed and before the user has access. Options to set target button text and icon as well as ability to add tool tips. Also included with this release is support for LOCAL compiles and CHM (compiled HTML help). A C7 (beta) install should be ready Monday. The C7 install, like all the upcoming C6 installs and eventual legacy support, will use the same upgrade unlock codes. RPM pricing is currently discounted 15%.

Posted Monday, September 17, 2007

BoxSoft Super Stuff 6.60

BoxSoft Super Stuff 6.60 is the first BoxSoft installation compatible with the new Clarion 7.0 beta. For upgrades and the new passwords please contact Mitten Software.

Posted Monday, September 17, 2007

XFiles 1.6

xFiles version 1.6 is available for download. This is a free upgrade to all current xFiles users. Support for VIEWS has been added. Since VIEWS support both ordering, and filtering, as well as giving you complete control over the fields, this is a major step forward if you wish to export subsets of files to XML. Support for simple SOAP packets has been added. An example is included (which uses NetTalk) which shows you how you build a SOAP client. A ValidateRecord method has been added, which allows you to do client-side filtering when exporting, or importing XML.

Posted Monday, September 17, 2007

Clarion Third Party Profile Exchange Adds Web Update

The Clarion Third Party Profile Exchange is now available in web update format. Product Scope 7, coupled with the Clarion Third Party Profile Exchange data version (web updated), allows you access to a wealth of Clarion Third Party add-on product information - plus a lot of other developer tools as well in extended paragraph descriptions about products and much more. For a limited time an introductory price of $24.95 is available.

Posted Monday, September 17, 2007

StrategryOnline Releases J-Media

StrategyOnline's J-Media is a Clarion control wrapper for the VLC Media Player's ActiveX control. VLC Media Player is small, open source, and supports most media formats as well as DVD playback.

Posted Monday, September 17, 2007

C7 Styled Menus Example App

Bob Z has posted a demo app showing how to change menu styles at runtime in C7. EXE and source are available.

Posted Thursday, September 13, 2007

C7 Project System Movie

Bob Foreman narrates this 12 minute movie covering details of the new C7 project system. Bob begins with the IDE start page and goes through the available ways to open new and existing projects. He covers the use of the Project Pad (which shows the project's components including source files), PRJ, CWPROJ, and SLN files, and demonstrates a multi-DLL solution which can be compiled all at once. When creating a new project you have the choice of Win32DLL, Win32EXE, or Win32 Library quick starts (AppGen will add new app types when it is released). Bob also covers the IDE's support for multiple versions of Clarion and some important and potentially confusing changes in project settings. Finally Bob covers some of the many configuration settings in Tools|Options.

Posted Thursday, September 13, 2007

C7 Visual Styles Movie

Bob Foreman narrates this six minute movie shows how to add visual styles and a manifest to a C7 app via the patched C6 templates included with the C7 beta. Bob covers the XP/Vista Manifest settings as well as the Extended UI settings. New UI changes include MDI tabs in a variety of styles. The movie also demonstrates opening/compiling a C6 project in C7.

Posted Thursday, September 13, 2007

Clarion Magazine

The New Dictionary Diagrammer

by Dave Harms

Published 2007-09-30

In a recent alpha build SoftVelocity included an early release of the new Dictionary Diagram tool. Visual design tools aren't new to Clarion; for years Enterprise Edition has come with the unfortunately-named PeaBrain Data Modeller. And while Data Modeller has its fans, most developers I know found it buggy and visually unimpressive.

The Data Diagram tool (hereafter called DD, although DDT has some appeal, assuming you see the diagrammer as a bug-killing aid) is a much more visually polished product than Data Modeller. On the other hand, DM lets you actually modify the dictionary, while DD in its current incarnation only creates diagrams. Down the road you can expect DD to add editing capability.

As with all the alpha/beta reports, keep in mind that the following information is subject to change.

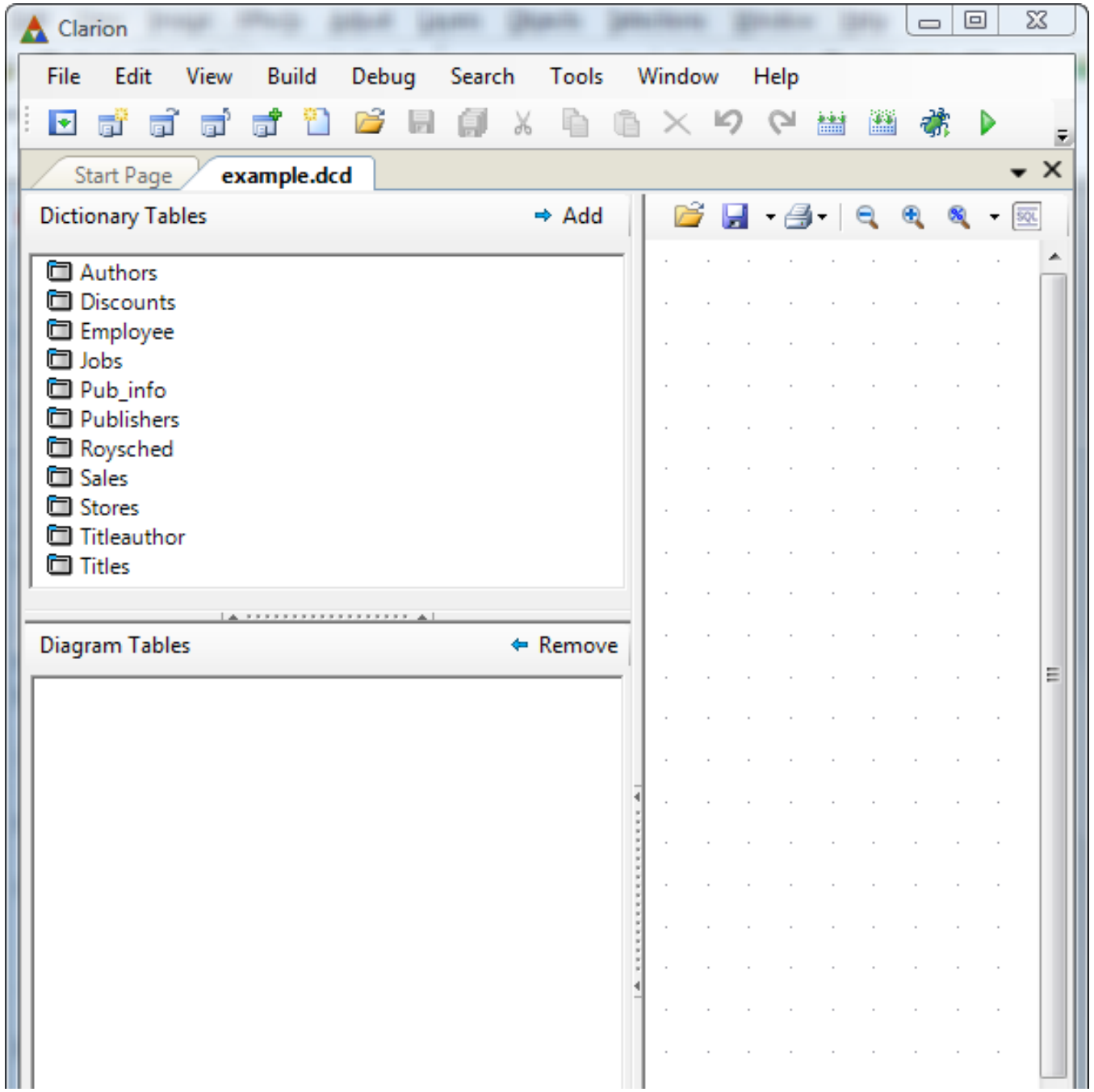
Creating a diagram

Creating a new diagram is a two step process:

1. Specify a filename for the diagram
2. Choose a data dictionary

In the alpha build I'm using you can only open or create a diagram from the start page - this will certainly change so I'll spare you the dialogs.

Figure 1 shows the IDE with a newly-created diagram, but with no tables yet drawn in the diagram. I'm using browses.dct from the ABC examples, I've switched to Plain layout, and I've squished down the IDE window so you can see the three diagrammer panes.



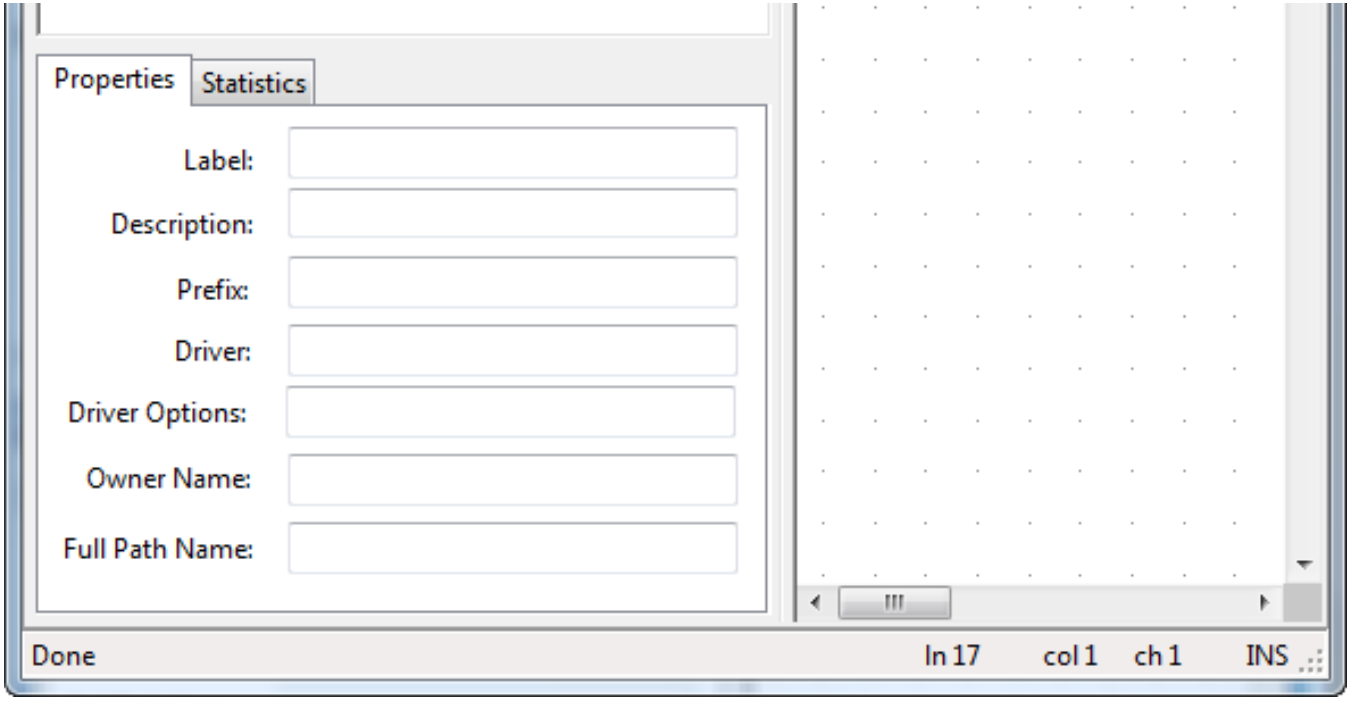


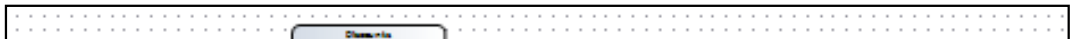
Figure 1. A new diagram with no tables added

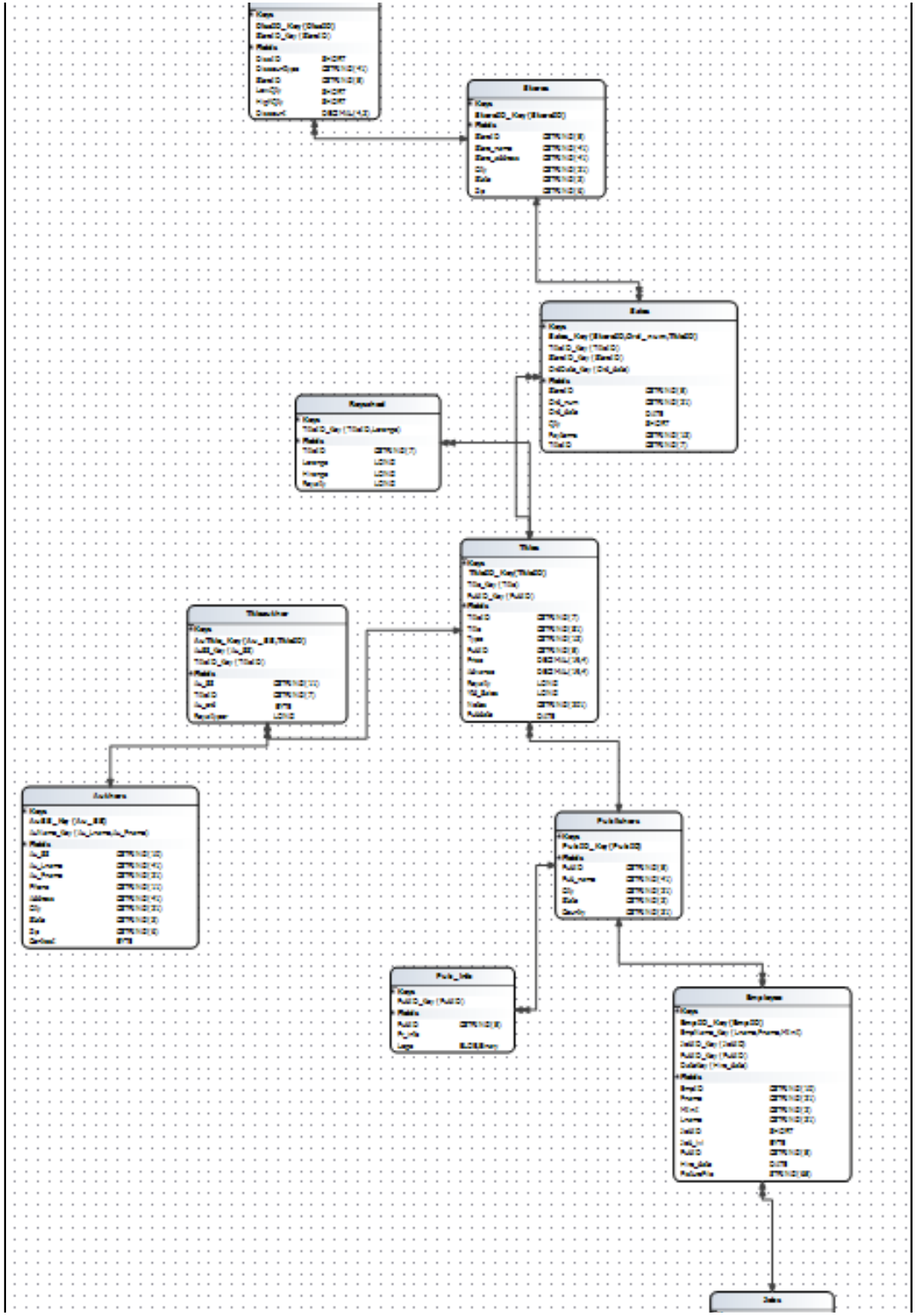
The top-left pane lists all of the files in the dictionary; if there are more than will fit in the pane you'll see a scroll bar, just as you'd expect.

The lower-left pane lists all of the tables in the diagram. This list functions as a locator - double-click on a table name to have the diagram zoom to that table. You can also see two tabs at the bottom for the currently selected table, one for basic table properties, and another for statistics including the number of fields, keys, and relationships, and the record size. The properties are, at present, read-only, but eventually you will be able to invoke the dictionary editor dialogs from the diagram and your changes will immediately be reflected in the diagram.

Adding tables to the diagram

Adding tables to the diagram is easy - you just highlight the tables you want (in the upper left-hand pane) and click on the Add button. In Figure 2 I selected all the files and DD took a stab at the layout.





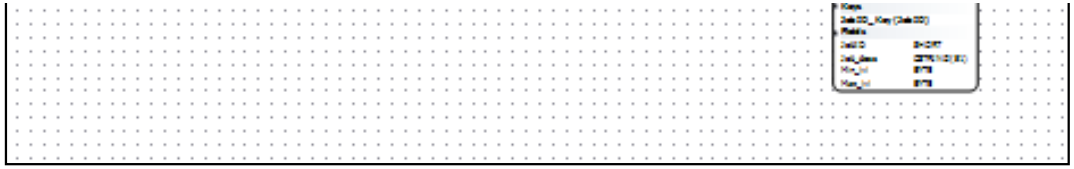


Figure 2. A default diagram layout (view full size image**)**

Actually when I clicked on Add all I saw was one tiny part of the diagram. DD's zooming and panning features are extensive; to get the view in Figure 2 I sized the panel and selected Fit All from the zoom menu.

Leaving aside for the moment the issue that the diagram is zoomed too far out to be readable, there are a couple of points to note in Figure 2. One is that all of the relationships are drawn in automatically; the other is that DD has done a fairly good job of placing the tables. Tweaking the layout is child's play - you just drag the tables around. Figure 3 shows the result after some very minimal shuffling.

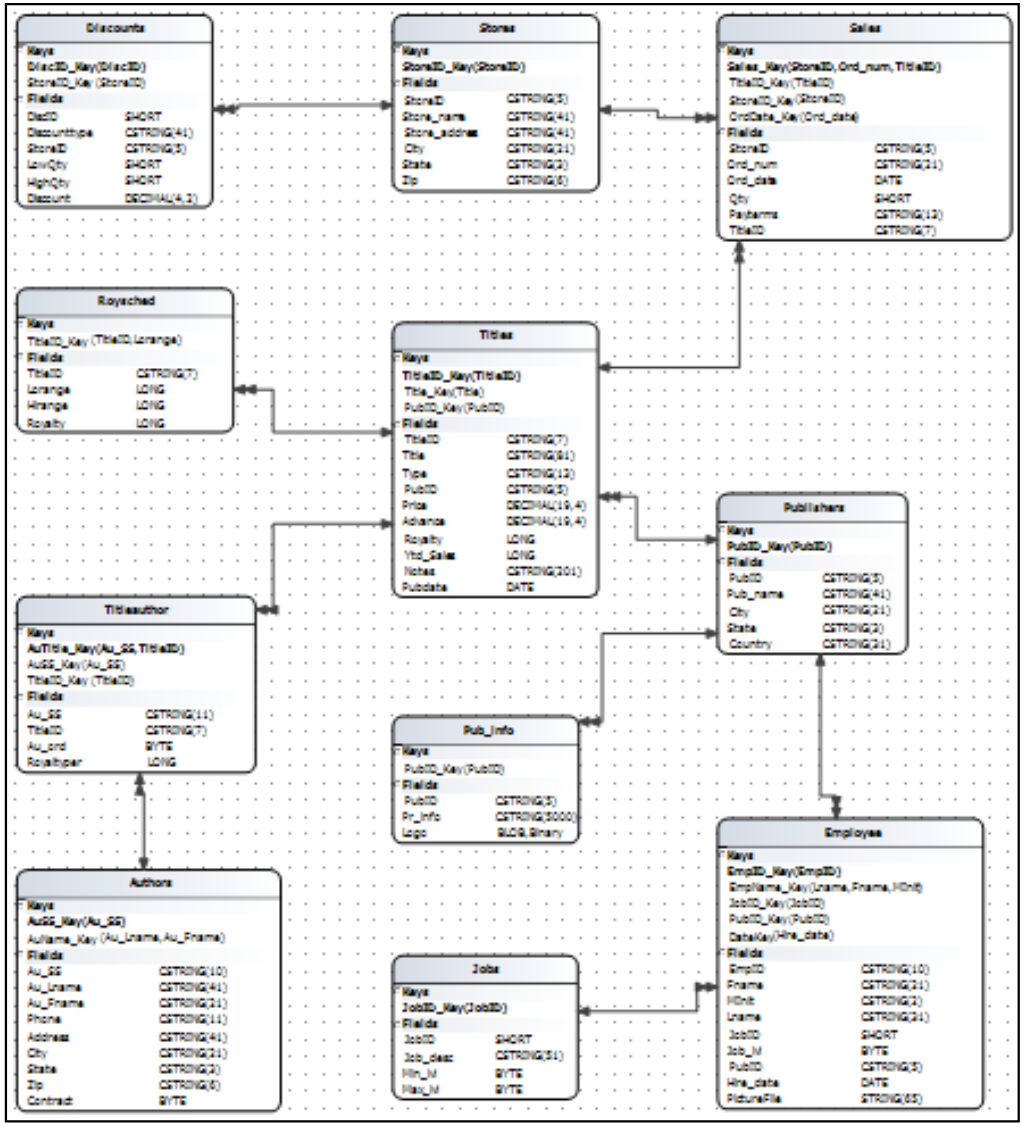


Figure 3. Rearranging the layout

There are a few features I'd like, such as the ability to drag the end points on the relationship lines and have them snap to points on the tables. But this is pretty slick for about a minute's work.

The diagram toolbar

Figure 4 shows the toolbar for the diagram pane.

Figure 4. Diagram pane toolbar

The first button brings up the Open dialog; the second gives you two options to Save. One is as a diagram file (DCD), the other is to save as an image file. Options are Windows Bitmap, GIF, JPEG, and PNG. The saved diagram looks like like the on-screen version, zoomed to 100%. You can view an example here.

The print button has three options: Print, Print Preview, and Print to PDF (this last option is not yet available). Choosing print preview brings up the window shown in Figure 5.

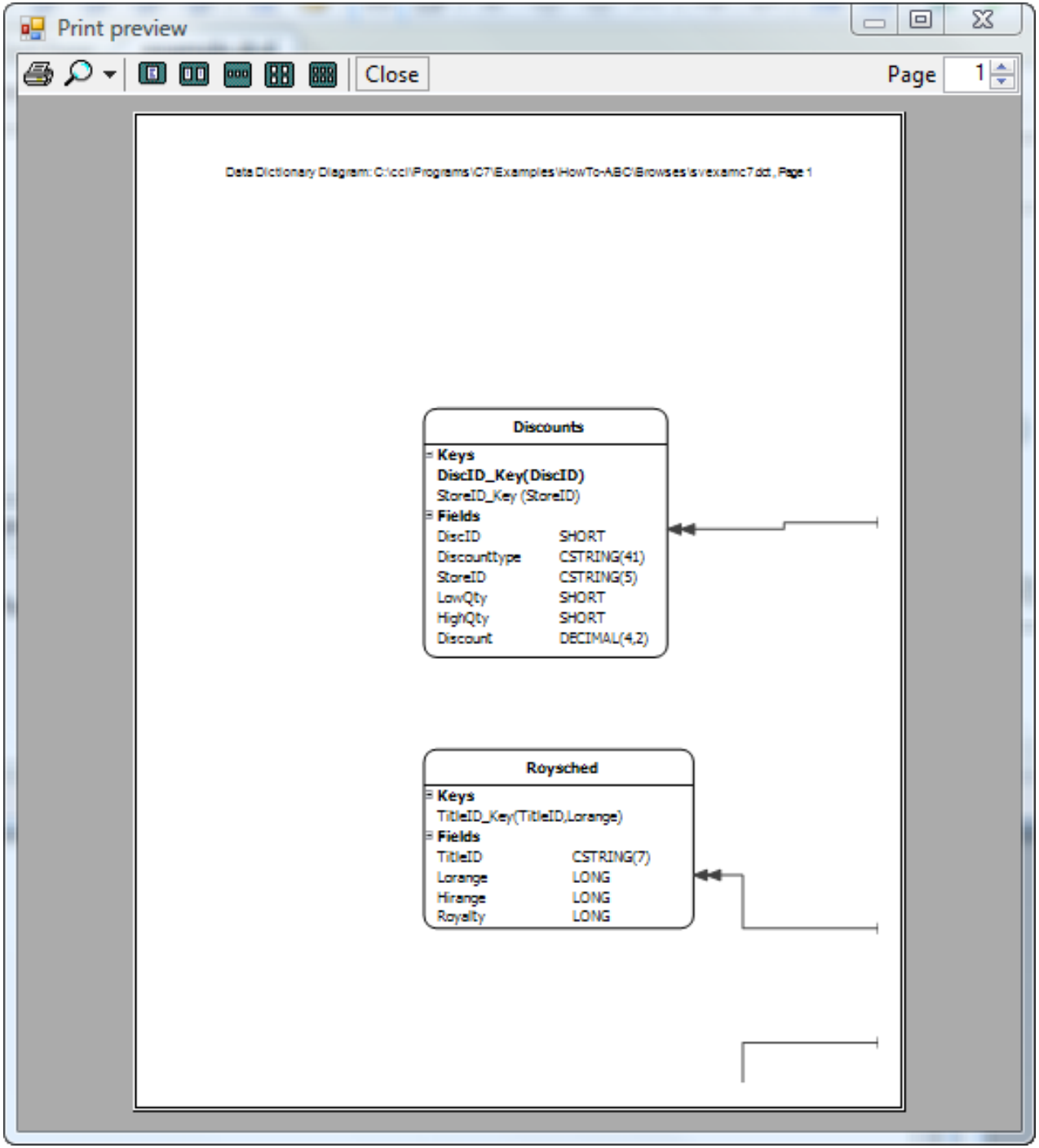


Figure 5. The print preview showing the first page of a diagram.

Choosing a six-page layout results in the preview shown in Figure 6.

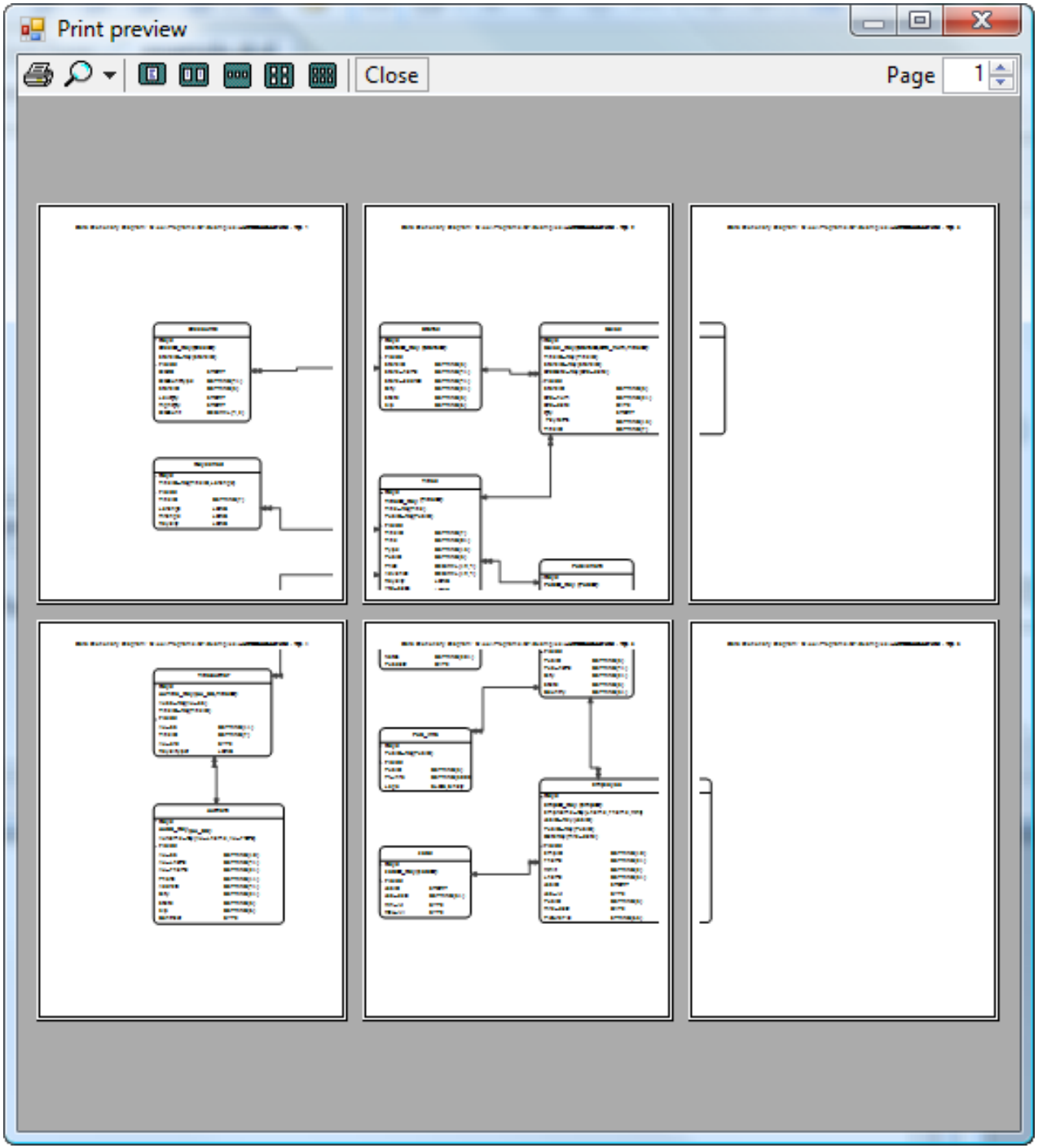


Figure 6. Previewing six pages at once.

The magnifying glass icons work as expected, letting you zoom in and out, but the third button has a number of useful menu options (see Figure 7).

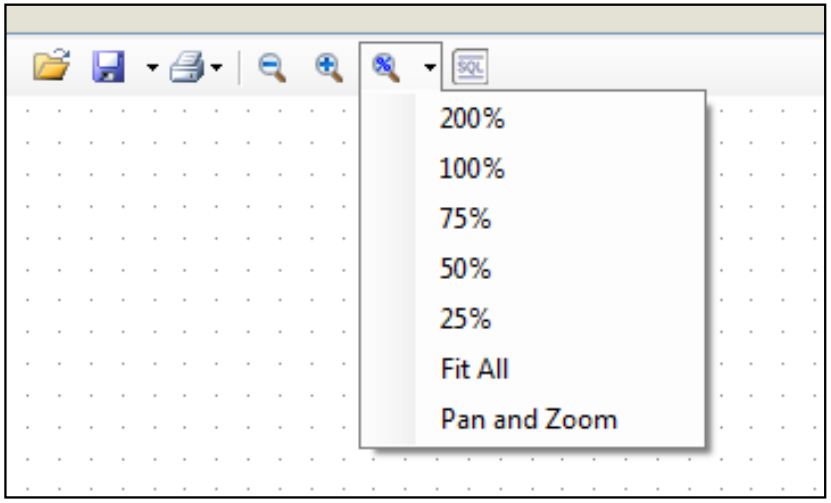


Figure 7. The zoom menu

Besides the various preset magnification settings there is a Fit All option which fits the diagram in the window. There is also a very useful Pan and Zoom tool which makes it easy to navigate around the diagram (Figure 8). Set the zoom slider to the desired magnification, and drag the view window around with the mouse.

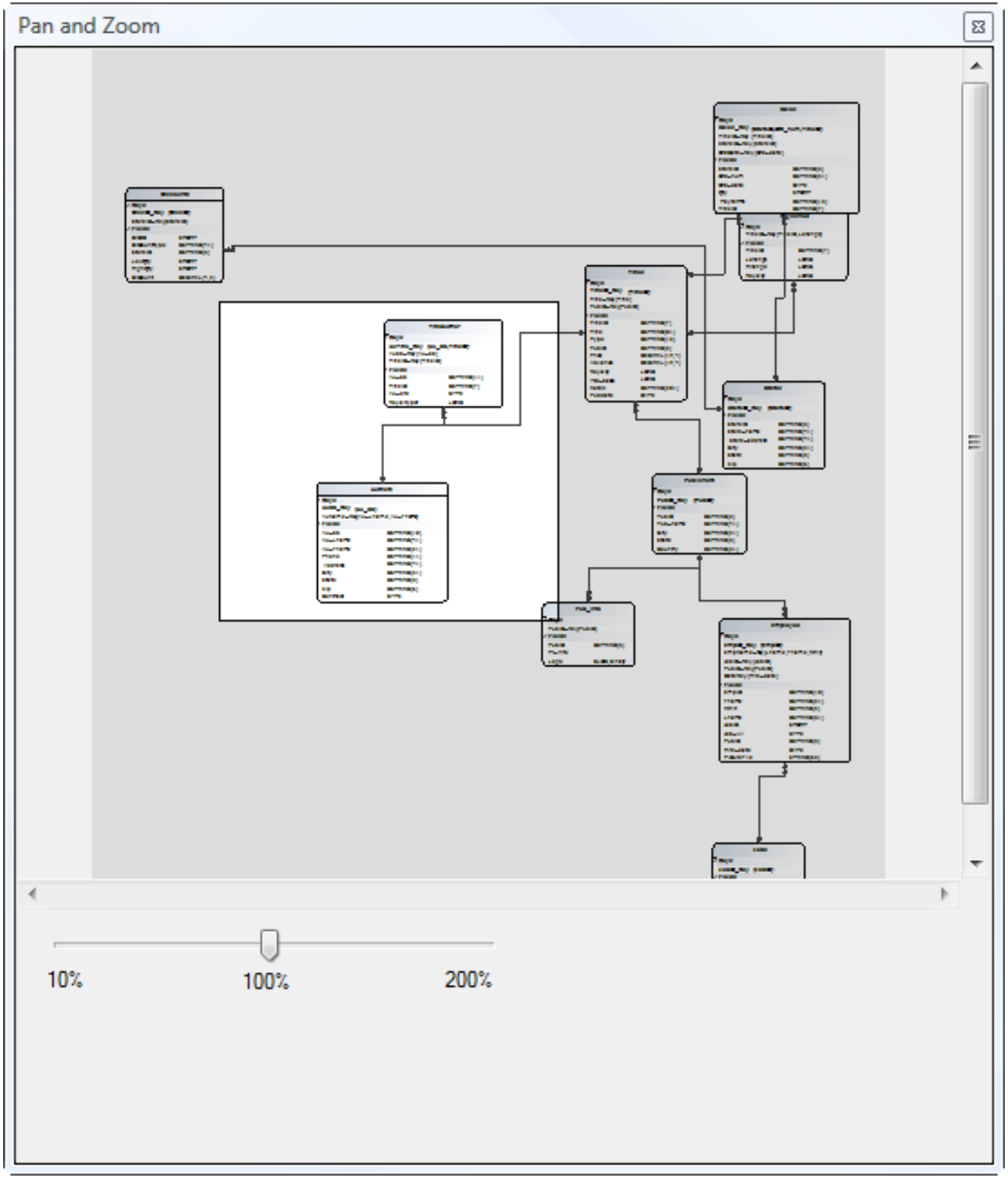

Pan and Zoom
10%
100%
200%

Figure 8. The Pan and Zoom tool

You can drag the Pan and Zoom window anywhere on your desktop, so if you wish you can leave it open outside the Clarion IDE entirely. You can work on the diagram at the same time, and use the Pan and Zoom controls to navigate. The Pan/Zoom window is resizable so if you're working with multiple monitors you may find it useful to have it displayed on one monitor while you work on the diagram in another.

The last item on the toolbar is disabled, but the tooltip says "Generate SQL". Presumably that means generate the SQL statements to create the tables.

The context menu

Click anywhere on the diagram background to get the context menu shown in Figure 9.

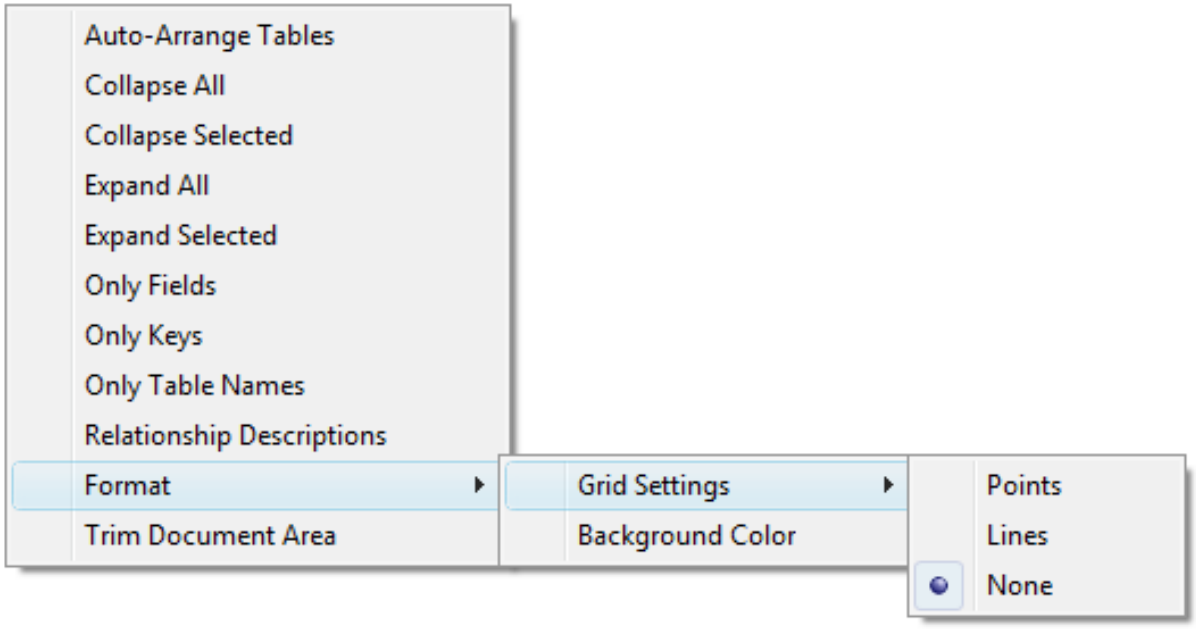


Figure 9. The context menu

The menu options are as follows:

- Auto-arrange - this will reshuffle all the tables according to the DD's layout algorithm. It works fine on smaller numbers of tables, but on a set of over 50 tables, with 30 or more related, the tables became a bit jumbled. This is a candidate for some further tweaking.
- Collapse all - only show the table name and the Keys and Fields headings (see Figure 10)
- Collapse selected - collapsed view only for the selected tables (drag a rectangle to select or use ctrl-click)
- Expand all - show all field/key data for all tables
- Expand selected - show all field/keydata for selected tables
- Only Fields - only show the fields, not the keys
- Only Keys - only show the keys, not the fields
- Only table Names - just the table names, nothing else
- Relationship Descriptions - show the descriptions on or alongside the relationship lines
- Format - change grid settings and background color
- Trim Document Area - remove unused space. Moving an object outside the diagram area increases the diagram's size.

Figure 10. Collapsed view

Diagramming large dictionaries

The C7 help file advises against diagramming large numbers of tables at once. Besides things getting a bit tight in the default layout, as in Figure 11, you'll find that performance suffers.

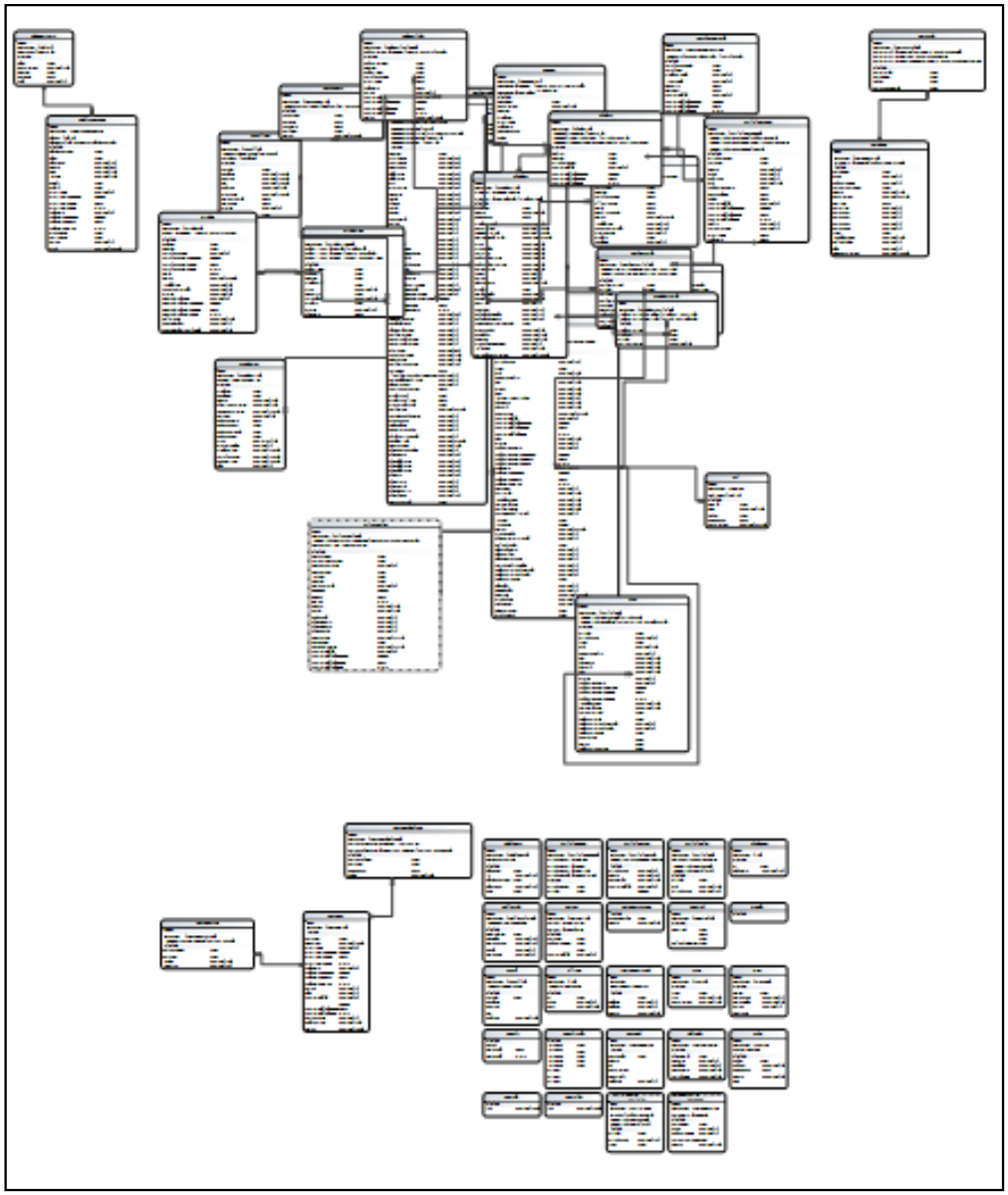


Figure 11. A default layout

Switching to displaying just table names (see Figure 12) improves performance and makes it far easier to identify tables and tease them apart manually. Once you have the layout sorted you can start making room for the field and table lists.

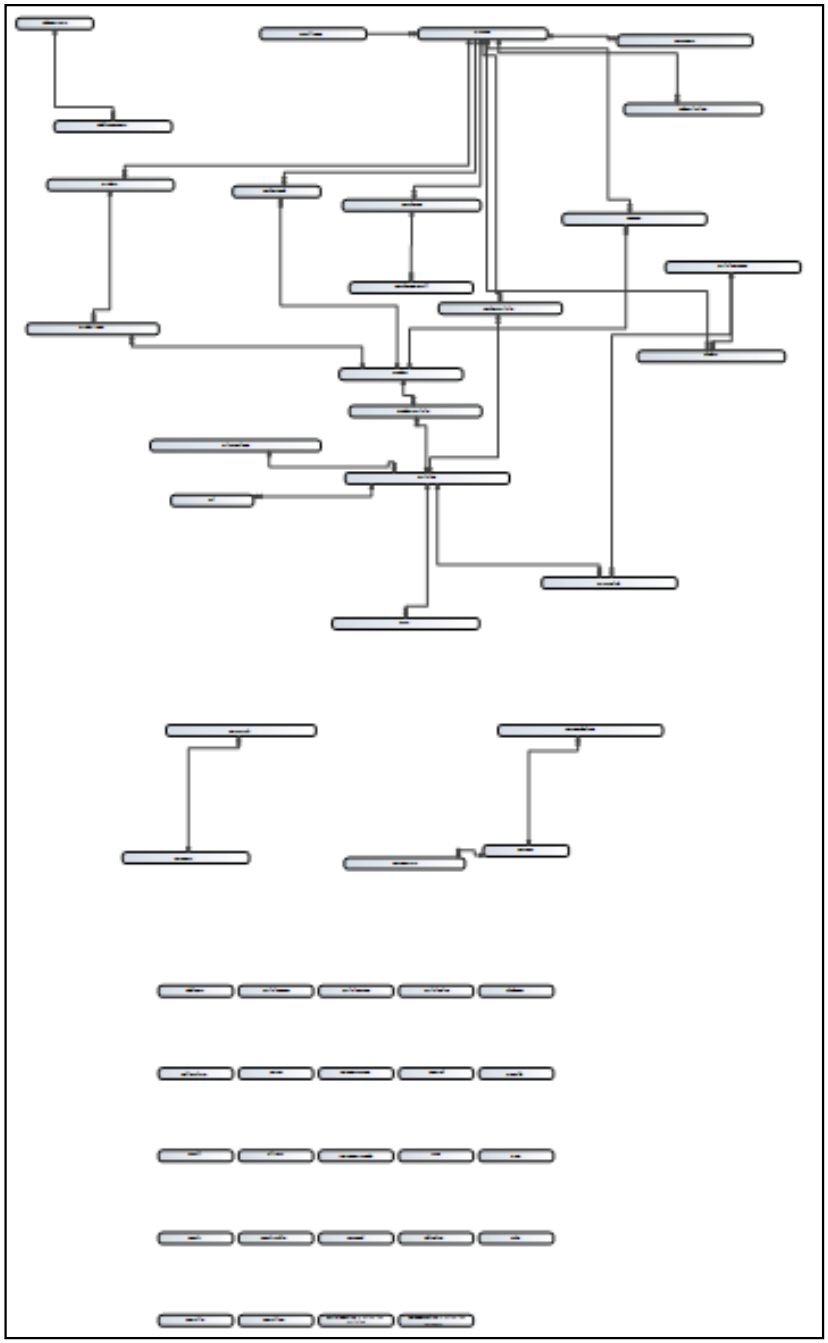


Figure 12. Rearranging with names only

Summary

The alpha release of the new Dictionary Diagram tool has a few rough edges, but is showing a lot of promise. I'm looking forward to being able to edit the dictionary directly from the diagram, but meanwhile the ability to diagram directly from a Clarion dictionary is something I'm sure many Clarion developers will appreciate.

Keep in mind that DD is an EE feature - it doesn't ship with Clarion Professional Edition.

Resources

- Bob Zaunere posted a video of DD back in August. There's no narration but there are on-screen notes to clarify certain points.

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

Reader Comments

Add a comment

Clarion Magazine

Clarion.NET Language Changes: What's New

by Dave Harms

Published 2007-09-27

In Part 1 I looked at the language elements that have been lost in Clarion.NET. While some of these changes are pretty significant, there's a lot of offsetting cool stuff to look forward to.

The short list of items that have been added to Clarion.NET includes:

- Multiple constructors with parameters
- Use NEW in expressions
- New INTERNAL attribute for classes, also some changes to access modifiers
- New data types (BOOL, CHAR, STRUCT, ENUM, references to arrays)
- New and changed assignment operators (:=, +=, -=)
- New type and casting operators (AS, IS, TRYAS)
- New class operators (DELEGATE, EVENT, PROPERTY, INDEXER)
- New control structure: FOREACH
- Exception handling (TRY, CATCH, THROW, FINALLY)
- Overflow checking (CHECKED, UNCHECKED)
- Critical sections made easier with SYNCLOCK (think sync lock, not syn clock)
- REF attribute for parameters passed by reference
- Variable method arguments in array (PARAMS)
- PROP:PrintDocument for reports and .NET Document objects
- Support for calling Win32 code via Pinvoke
- NAMESPACE and USING

Multiple constructors and NEW in expressions

Multiple constructors add a lot of flexibility to object-oriented programming by giving you more choices as to how you initialize a new object. With multiple constructors you can start the object in various states without having to call additional methods. Here are three possible ways you could initialize a class with three constructors:

```
myObject = new MyClass()
myObject = new MyClass('some text')
myObject = new MyClass(123.65)
```

Of course you can accomplish the same thing by creating an object and then calling one or more methods, but the new syntax

is a great help if you want to cleanly do something like the following, which appears to be allowable syntax now that NEW can be used in any expression:

```
objA := new A(new B(83.2))
```

Access modifiers

Clarion.NET's class access modifiers are the same as C#, as follows:

PUBLIC - accessible to all code in any assembly
INTERNAL - accessible to all code inside the same assembly (default)
PROTECTED - accessible only to code of classes that extend the class where the named entity exits
PRIVATE - only accessible to code that belongs to the same class

As Carl Barnes has pointed, these modifiers are also the same as VB except Internal is Friend.

Remember that .NET applications internally are made up of one or more assemblies; a DLL or EXE is an example of an assembly.

New data types

The BOOL (boolean) and CHAR (unicode character) data types are new, and actually a bunch of others are too, including STRING, DATE, TIME, in other words all the types that now have CLA*originalname* declarations. These are .NET data types, and you'll have the choice of sticking with the .NET standards or using the Clarion.NET enhanced versions.

Clarion.NET sports STRUCTs and ENUMs. A STRUCT can have data and methods and looks much like a class but is a value type, not a reference type. The help file lists the following properties for STRUCTs:

- STRUCTs are value types while CLASSes are reference types.
- When passing a STRUCT to a method, it is passed by value instead of as a reference.
- Unlike CLASSes, STRUCTs can be instantiated without using a new operator.
- STRUCTs can declare constructors, but they must take parameters.
- A STRUCT cannot inherit from another STRUCT or CLASS, and it cannot be the base of a CLASS. All STRUCTs inherit directly from System.ValueType, which inherits from System.Object.
- A STRUCT can implement interfaces.

An ENUM looks like, and does the same job as, the ITEMIZE structure. However ENUM is more than a structure; it's a .NET class with a number of useful methods.

Array references

Clarion.NET adds support for array references. For instance, here's a three-dimensional array of LONG value:

```
myArray long[,,]
```

The help file also provides the following example of "a reference to an array of references to a LONG with 1 dimension":

```
ArrRefLg &LONG[]
```

Note that in the first example no & is used.

All of the new data types are in fact classes, which raises an important point: although Clarion is a hybrid language, both procedure and object-oriented, .NET is strictly an object-oriented platform. Where does Clarion's procedural code fit in?

Clarion.NET - it's all OOP

Although a typical Clarion.NET application may still look like a mix of procedural and object-oriented code, under the hood it's all OOP, baby. Every variable is an instance of a class (however small it might be), and every procedure is...a method. That's right, a method. You may not see the method's class - you don't have to - but it's still there, lurking in the background. You just see the one method, which you think of as a procedure. It's all done with smoke and mirrors.

Assignment operators

There is a new "smart assignment" operator, := (colon-equals), which works the same as = and &=. The := operator does a simple value assignment like = where needed, and a reference assigment like &= where that is appropriate.

Clarion.NET also adds enhancements to the increment and decrement operators. In addition to math functions these are now used for registering/unregistering event handlers.

Type checking and casting

Type checking is determining if an object has a specific class type, and casting is the process of converting an object of one type to an object of another type. For type checking Clarion.NET has the IS operator. For converting an object from one type to another you use the AS and TRYAS operators. TRYAS throws an exception if the cast fails.

You can also use the TYPEOF function to get the type (that is, the class) of any object.

New CLASS attributes

There are a number of changes to CLASSes in Clarion.NET, most significantly support for events and delegates. A delegate is a sort of object-oriented type-safe callback, and delegates are widely used in responding to events. You can have multiple delegates registered for a given event.

This is the technology that's taking the place of the ACCEPT loop in Clarion.NET. There's a whole lotta shaking going on here, and a meaningful discussion of delegates is well beyond the scope of this article. I'll have more to say about these and other CLASS changes in a future article.

Looping with FOREACH

FOREACH is a control statement that lets you loop once through each element in an array (or, presumably, any list). See the TRY/CATCH example below.

Exception handling

As Clarion developers we're all familiar with error handling - we execute some code, check for an error, and respond if necessary. In .NET (and in languages like Java) it's possible to do something that's a little more like a database transaction. You mark off a block of code and if anything goes wrong anywhere within that code it terminates and you can do something about the error. In Clarion.NET, as in other .NET languages, a block of code like that is marked with a TRY structure, followed by a CATCH structure to respond to any error, and optionally a FINALLY section. The code in FINALLY will always execute whether or not there was an exception. Here's an example (which is a munged version of something that's in the Clarion.NET help):

```
TRY
  FOREACH ctrl in SELF.Controls
    IF(ctrl Is TextBox)
      IF(NOT ctrl.Tag&=null)
        colnm&=ctrl.Tag
        tab&=SELF.m_Row.Table
        tp&=tab.Columns[colnm].DataType
        val&=(ctrl TryAS TextBox).Text
        IF((ctrl TryAs TextBox).Modified)
          THROW NEW DataException('Data error: ' & ERROR())
        END
        SELF.m_Row[colnm]:=Convert.ChangeType(val,tp)
      END
    END
  END
CATCH Exception e
  MESSAGE('Dang it, something"s wrong:' & e.Message)
FINALLY
  MESSAGE('Done - time to clean up')
  ! Cleanup code (if any) goes here
END
```

As the above example shows, you don't have to wait for library code to throw an error; you can trigger an exception with the THROW statement.

Overflow checking

The CHECKED and UNCHECKED operators control overflow checking for integer math operations.

Critical sections

The new SYNCLOCK procedure executes the expression passed to it inside a critical section. For more on critical sections see the threading subtopic.

REF attribute

As in C#, the REF attribute specifies parameters passed by reference, and is added for compatibility with other .NET languages

Variable method arguments

In Clarion.NET you can pass an array of parameters to a method. Here's an example from the help:

```
TOTAL PROCEDURE(PARAMS LONG [] VARARG), LONG
```

You then reference VARARG as an array of type LONG. The MAXIMUM value of the array is equal to the number of array elements passed in. Many languages use this technique when passing parameters on the command line.

Report previewing

The new PROP:PrintDocument property lets you attach a report structure to a .NET PrintDocument object so the report can be previewed in any .NET print previewer. This is one small example of how Clarion.NET leverages the power of the vast .NET class library.

Calling Win32 code

The .NET framework provides a means of calling Win32 code, and Clarion's support includes both .NET's DLLImportAttribute attribute and the more familiar Clarion MAP syntax. Declaring a Win32 procedure in Clarion.NET looks very much like it does in Clarion Win32.

NAMESPACE and USING

As a .NET language, Clarion.NET conforms to the requirement that all classes must be in a namespace. If you don't specify a namespace with NAMESPACE, Clarion will add one for you. The USING directive lets you use the classes in that namespace without specifying their fully qualified name (*namespace.classname)* where namespace may really be a namespace hierarchy. For instance, if you want to call the System.Drawing.Printing.PrintDocument class, adding a USING directive for System.Drawing.Printing will allow you to refer to the class as just PrintDocument.

Summary

In these two articles I've touched on the language changes in Clarion.NET, as they're described in the C7 beta help file. No doubt there will be more changes on the way to gold release.

While some of the changes, such as the loss of unsafe functions like ADDRESS and the changes to window handling, will make code conversion a bit of a challenge, Clarion.NET looks to offer a workable upgrade path from Win32 code. And I expect that anyone capable of getting down and dirty with Win32 code will find .NET provides even better tools and resources.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995), and has written or co-

written several Java books. David is a member of the American Society of Journalists and Authors (ASJA).

Reader Comments

Add a comment

Clarion Magazine

Clarion.NET Language Changes: What's Gone

by Dave Harms

Published 2007-09-27

For those of us waiting anxiously for Clarion.NET there is some useful information to be had in the C7 beta release. Since both C7 and Clarion.NET use the same IDE it appears that there will be one help file covering both products, and there is already a fair bit of information in the new help file on Clarion.NET. In this article I'll cover some of the Clarion.NET language changes as listed in the help as well as some of the structural differences between the Win32 and .NET Clarion languages.

What's so different about .NET?

I've discussed .NET vs. Win32 architecture issues in previous articles. In .NET Basics: What Is .NET, And Why Should I Care? I gave a very brief history of the evolution of programming from Windows API to .NET platform, and in Part 2 I took a closer look at the vast library of code that ships with the .NET platform, and which will be readily available to programmers using Clarion.NET. In .NET for Clarion Developers I compared managed with unmanaged code, touched on changes to window handling, datatypes and design conventions, and very briefly considered web development and mobile platforms. I suggest you have a look at those articles before proceeding.

Please note that in this article I'm focusing on *language changes only* and not on the supporting code, including ABC, templates, drivers, etc. Also keep in mind that Clarion.NET isn't even in alpha/beta yet, and although I've had confirmation of a number of the following points from SoftVelocity you can expect changes between now and final release.

As a look at the new help file shows, a great deal has changed in Clarion.NET; at the same time SoftVelocity's stated goal is to make the transition from Clarion to Clarion.NET as easy as possible. So what's gone missing, and what's been added? First, here's the stuff that's gone.

Gone in Clarion.NET

The short list of items removed in Clarion.NET includes:

- The WINDOW structure (and window-related statements like DISPLAY, UPDATE etc.)
- The ACCEPT loop
- The REPLACE attribute in constructor methods
- Option to use only some array indices
- The deep assignment operator
- The OVER attribute for variable declarations
- The ADDRESS method

In the following discussion I've loosely grouped these language changes by the reasons for their removal, which include WinForms, .NET language restrictions, and code safety.

WinForms

We're all used to Clarion's WINDOW structure, but .NET comes with an extensive window handling library for desktops applications, called Windows Forms, or WinForms. One of the decisions SoftVelocity had to make was how complete to make Clarion.NET's support for WinForms. Would they continue with the proprietary WINDOW structure and add some sort of translation layer, or go native? In the end they went native, and while that will mean some pain in the conversion process I believe it was the right choice. The .NET platform contains not just code for application developers, but for tool makers; by aligning Clarion more completely with .NET, SoftVelocity can take better advantage of that code.

The move to full WinForms support is also the reason the ACCEPT loop is gone (in favor of standard WinForms event handling), but that change needn't be as radical as it seems. In an ACCEPT loop you essentially have a black box, the ACCEPT statement, which traps certain events and then lets you execute code for those events. In WinForms events also trigger calls to code, except that each event gets its own method (if you're not yet conversant with OOP, a method is really just another kind of procedure). The WinForms approach is a little less centralized, since event handler methods are associated with individual controls rather than all funnelled through the window, but otherwise there isn't much functional difference. Events still trigger code. And it's not that different a model from ABC anyway, since the ABC classes hide the ACCEPT loop and your embed code actually sits inside a virtual method.

The loss of WINDOW-related statements like DISPLAY, UPDATE and many others may be more of a headache, but in each case you can expect to see a replacement WinForms method to call.

REPORT structures, by the way, are the same in Clarion.NET (there's no comparable library in the .NET platform), so reports should be relatively easy to port.

.NET language requirements

There are a number of language requirements that .NET imposes on Clarion, including constructor/destructors and array handling.

The REPLACE attribute is gone in Clarion.NET, at least according to the current documentation. One of the things you'll notice in Clarion.NET is the ability to use of multiple constructors (see Part 2). And when you have multiple constructors you have the choice of which constructor to call, and one constructor can call another constructor (although it has to do this before any other code in the method). So in Clarion.NET rather than use REPLACE to put in your own code you'll call the base class constructor from your constructor, then run any additional setup code.

There are some changes to array handling. You can still have multi-dimensional arrays in Clarion.NET, but now you have to specify all elements of the array. Here's an array declaration as taken from the Clarion help:

```
Scr   GROUP             !Characters on a DOS text-mode screen
Row     GROUP,DIM(25)   !Twenty-five rows
Pos     GROUP,DIM(80) !Two thousand positions
Attr      BYTE        !Attribute byte
```

```
Char       BYTE       !Character byte
        END
      END
    END
```

In Clarion Win32:

- Scr is a 4,000 byte GROUP
- Scr.Row is a 4,000 byte GROUP
- Scr.Row[1] is a 160 byte GROUP
- Scr.Row[1].Pos is a 160 byte GROUP
- Scr.Row[1].Pos[1] is a 2 byte GROUP
- Scr.Row[1].Pos[1].Attr is a single BYTE
- Scr.Row[1].Pos[1].Char is a single BYTE

However, in Clarion.NET only the last three (presumably) are legal because all index arrays must be specified.

Interestingly, one feature that was due to be scrapped in Clarion.NET has survived: the . shorthand for END. Clarion Magazine surveyed developers on this subject a few years ago and the response was mixed. Clearly the loss of . would have bitten a number of developers in the shorts, and those developers can now relax.

Code safety

In a pure .NET application the .NET runtime manages the application's running code, preventing nasty things like GPFs. As the size of applications and the power of programming languages has grown, it's become even more important to regulate just what a program can and cannot do. For Clarion developers the move to managed code may seem like a throwback. Back in the DOS days, Clarion Professional Developer created pseudocode that executed on a runtime interpreter, and there was no easy way to use pointers or otherwise directly access memory. In Clarion.NET, the Clarion compiler creates IL (Intermediate Language) code which is executed by the .NET platform, and there is no easy way to use pointers or otherwise directly access memory.

One of the casualties of the move to .NET managed code is the loss of language statements that are inherently unsafe (that is, they have the potential to access memory in an uncontrolled manner). These include OVER, ADDRESS, and the deep assignment operator.

ADDRESS is obviously going to get the boot, since in .NET you deal in references to objects, not absolute addresses. Some use of ADDRESS has to do with working around restrictions on casting and passing around objects, and I suspect many of those restrictions will be gone in Clarion.NET. But if you're using ADDRESS to manipulate pointers, you're definitely going to have to rethink your code. You will probably lose some performance, but in general that's the case anyway with going to the .NET platform. Code management comes at a price. If you absolutely have to manipulate memory directly you can always put that code in a library and include it as unmanaged Win32 code.

OVER is a less-obvious casualty of the restriction on direct memory access. One common use of OVER is to convert a string into an array of strings, like this abridged example from the ABC report target code:

```
LOC:Buffer      STRING(4)
```

```
GG          GROUP,OVER(LOC:Buffer)
A           BYTE
B           BYTE
C           BYTE
D           BYTE
          END
G           GROUP,OVER(LOC:Buffer)
A           BYTE
B           BYTE
C           BYTE
D           BYTE
          END
LOC:NBuffer    ULONG,OVER(G)
```

In this code, both the GG group and the G group occupy the same memory space as LOC:Buffer, a four byte string. As well, LOC:Nbuffer, a four byte LONG, occupies the same space as LOC:Buffer. In .NET you'll most likely wrap this kind of code up in a class so you can manage the different ways you want to access a given piece of information. The .NET System namespace contains many classes and structures suitable for this kind of data manipulation.

Similarly, the deep assignment operator, which lets you copy data from one structure to another, is unsafe. As the help notes, "deep assignment is a structure piercing operation" which, among other things, that it lets you copy data from a group to a class, violating the class's encapsulation; you can overwrite private data (which suggests another workaround for ABC classes with inaccessible data...). Deep assignment matches structures only on level and label; it does not work through "normal channels" except that it makes use of automatic data conversion when matching structure elements have different data types.

Although deep assignment is missing from Clarion.NET it doesn't appear to be an impossible task to get it back. Last year Alan Telford wrote an article for Clarion Magazine describing a customized deep assign function using WHO and WHAT; since .NET lets you examine structures in even more detail than that allowed by WHO, WHAT and WHERE, it should be relatively easy to create a .NET equivalent to Clarion's deep assignment.

Clarion data types

Many Clarion data types don't have obvious .NET equivalents, so SV has created new classes to model these types. Here's the list:

Clarion WIN32	**Clarion.NET Equivalent**
DATE	CLADATE
DECIMAL	CLADECIMAL
LONG	CLALONG
STRING	CLASTRING
TIME	CLATIME

ULONG	CLAULONG

What this means is that anywhere you have the Win32 types in your source code you'll need to change the declaration to the Clarion.NET version. SV's promised conversion code will presumably take care of that for you.

So why didn't SV stick with the original data type names instead of renaming them? To avoid confusion with the standard .NET data types of the same name. This will matter a lot when developers begin porting, say, C# examples to Clarion.

So far I've looked at the downside changes to Clarion.NET. But there's a lot of cool new stuff, and I'll go into that next time.

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

Reader Comments

Posted on Monday, October 01, 2007 by Bob Sutton

Dave, does the phrase "loss of . would have bitten a number of developers in the shorts" imply that there will be a conversion utility/aid for existing code/apps?

.....

Posted on Tuesday, October 02, 2007 by Dave Harms

Bob,

Conversion tools are mentioned in the roadmap: "Migration to WinForms is eased by new Template support to convert Window structures to WinForms, as well as additional tools to convert Windows that are not contained within the .APP format into WinForms."

Dave

Add a comment

Clarion Magazine

Debugging SQL Problems With File Callbacks

by Marty Honea

Published 2007-09-19

Working in Clarion has been a blessing and a curse. One of the blessings is that many of the menial tasks required to make a useful program are handled by the standard Clarion templates. Ironically, that's also one of the curses. When these menial tasks are performed for us we sometimes lose sight of what's really going on under the covers. And when the behavior we've become accustomed to becomes behavior we don't want, it can be difficult to even know where in the standard code to look for a solution.

I recently had a situation that made it necessary for me to pull back the covers on one of my applications and figure out what was happening at a level that I normally take for granted.

A random SQL error

One of the programs I had inherited was failing on seemingly random SQL calls with an Entry Not Found error. This was a rather large program, and I couldn't reliably duplicate the error.

Almost all of the Clarion programs I work with on a day to day basis connect to some kind of SQL database. DebugView, the Clarion trace utility, WinSQL, and even the dreaded STOP and MESSAGE statements have become integral parts of my problem solving toolbox. But in this case these tools didn't provide enough answers. I needed a way to know what was happening at the file driver level, and I needed to be able to display values for variables in my application before and after certain events happened. This put me on a search for a better way to trace my program's interaction with the SQL back end.

File callbacks

After searching the Clarion help and getting some useful redirection from Bruce Johnson, I had a starting point: Clarion's CALLBACK function, which installs a file driver callback procedure.

There is also a SQLCallBack function, but according to the Clarion documentation it is a function that is only supported by the SQL drivers. By using a more generic function, I felt that I could reuse the my code in more situations.

Wikipedia defines a callback as "executable code that is passed as an argument to other code. It allows a lower-level software layer to call a subroutine (or function) defined in a higher-level layer." Clarion's CALLBACK lets you plug in your own code at specific points during file activity.

All I needed was to eavesdrop on the data going through the driver; I didn't need to take over handling all of the driver's events. Luckily, for me at least, more than one callback function can be registered at one time for an entity. What this means is that I can insert code to trace the variables I need to monitor without having to take over processing all the events I'm

not interested in.

Registering the callback function turns out to be pretty simple. Here's the prototype:

```
CALLBACK(entity, FileCallBackInterface, [flag])
```

The entity is the file or view for which you want to register your function, and FileCallBackInterface is an INTERFACE, which is sort of like a class definition that you have to implement yourself. I'll get to that in a moment. Passing a value of True in the optional flag parameter will unregister the function, and omitting the flag will register the function.

Events that are reported by the function are listed in the equates.clw file in your Clarion libsrc directory and have a prefix of DriverOp. An event will fire before a driver operation and after it has completed. There is a ton of information that can be reported, so limiting that information is very important. About the only thing as bad as no information when trying to debug a program is being snowballed with so much data that you can't find anything.

A few of the DriverOp events that you can trap are:

- DriverOp:Open
- DriverOp:CLOSE
- DriverOp:Add
- DriverOp:Put
- DriverOP:SetProperty

There are other events, but these cover the majority of what I want to check for when I'm tracking a file driver issue. The first four are pretty self explanatory, but the last one, DriverOP:SetProperty, is one that I find most helpful when using SQL statements because it's the event that's triggered when you execute a Filename{Prop:SQL}.

The interface and the class

Here's the definition of the INTERFACE which any class used in a callback must implement:

```
FileCallBackInterface INTERFACE
FunctionCalled        PROCEDURE(SIGNED opCode, |
                        *Params Parameters, |
                        *CSTRING FileErrCode, |
                        *CSTRING FileErrMsg), BYTE
FunctionDone          PROCEDURE(SIGNED opCode, |
                        *Params Parameters, |
                        *CSTRING FileErrCode, |
                        *CSTRING FileErrMsg), BYTE
                    END
```

You can find the declaration in libsrc\filecb.inc.

I've written a class called MHCallbackClass that implements the methods defined in FileCallBackInterface and makes it easy to use callbacks with multiple files. The class uses the db.Message method from freely-available DebugView Class

and Template to write to a system log, which you can view with Microsoft's (formerly SysInternals') DebugView. The class and template are based on Skip Williams' Debuger (see Debug De Program With Debuger), and you can easily put db.Message calls anywhere you like in your embed code which makes it much easier to find out which code is causing a particular driver event.

Here's the declaration for MHCallBackClass:

```
MHCallBackClass  Class,Implements(FileCallBackInterface), type
MyFile           &FILE,PROTECTED
MyFileName       CSTRING(FILE:MaxFilePath + 1)
Usage            UNSIGNED,PROTECTED
fileaction       CSTRING(41),PROTECTED
actionlevel      long,PROTECTED
errcode          CSTRING(FILE:MaxFilePath + 1)
errmsg           CSTRING(FILE:MaxFilePath + 1)
Opcode           Signed
Init             PROCEDURE(FILE aFile),VIRTUAL
Kill             PROCEDURE,VIRTUAL
GetLastAction    Procedure(unsigned opcode),Virtual
                 end
```

There are two methods in MHCallbackClass which are not listed in the class definition because they are specified in the INTERFACE which the class IMPLEMENTS: these are FunctionCalled and FunctionDone. FunctionCalled is called before an event occurs, and FunctionDone is called after an event has completed. opCode is the parameter that specifies the event being reported, and Parameters is a group declared with the Params type that gets passed with information pertaining to this event.

With the FunctionCalled method care needs to be taken when testing for error codes. Since the event you're monitoring hasn't occurred yet, if you have an errcode it is from the previous action taken.

Here's an annotated version of the source for FunctionCalled:

```
Myclass.FileCallBackInterface.FunctionCalled|
                        PROCEDURE(|
                          SIGNED opCode, |
                          *Params Parameters, |
                          *CSTRING ErrCode, |
                          *CSTRING errmsg)



!Params GROUP,TYPE
!Ahead      UNSIGNED
!Behind     UNSIGNED
!Buffer     UNSIGNED
```

```
!Field      LONG
!FieldList   &STRING
!Fields      LONG
!File       &FILE
!Index       SIGNED
!Key1        &KEY
!Key2        &KEY
!Len         UNSIGNED
!openMode    SIGNED
!Pointer     LONG
!Position    &STRING
!Property    &STRING
!Records     LONG
!Seconds     SIGNED
!Start       ULONG
!State       LONG
!Stop        ULONG
!Text        &STRING
!TimeOut     UNSIGNED
!ReturnStr   &STRING
!ReturnLong  LONG
!       END
RetVal  BYTE
 CODE

 If Errorcode()
    Self.errcode = Errorcode()
    Self.errmsg  = choose(errorcode() = 90,FileError(),Error())
    Self.GetLastAction(opCode)
    Self.Opcode = opCode
 Else
    Clear(Self.errcode)
    Clear(Self.errmsg)
    Return True
 END
 CASE opCode
 OF DriverOp:Add
    DB.Message('debug text here - see source code')
 OF DriverOp:AddLen
```

```
    DB.Message('debug text here - see source code')
  OF DriverOp:Append
    DB.Message('debug text here - see source code')
  !--- messages for a whole bunch of other opcodes ---
  END
  RETURN True
```

The parameters group that gets passed to a procedure is populated with information specific to the event that triggered the procedure. A list of valid parameters can be found by looking in the Clarion online help for "FileCallBackInterface and DriverOps", and as the listing shows I've also left a commented layout of the Params group in the Callbackclass.clw file for reference when working with the class. DB is an instance of the Debuger class,

Using the class

To use this class you include two files and instantiate a new class.

Callbackclass.clw gets included in the global program procedures section.

Callbackclass.Inc gets included in the global file declarations section.

You'll have to instantiate an instance of the class for each table used. In my application, for the members file I declare this class.

```
Memclass   class(MHCallbackClass)
        end
```

You'll have to initialize the class, passing it the file, before the file is opened.

```
Memclass.init(members)
```

After the file is closed you can kill the class

```
Memclass.Kill()
```

One cool thing about using file callback functions is that if you initiate a callback in one place in your application, accessing the file anywhere in that program will trigger events in your function as long as it is on the same connection, until you un-register the function.

By using this class I was able to find an errant close of a dummy file we use for calls to the database. Since I had logged which procedure and which routine each call was made from, I was able to quickly get the application back to behaving normally by removing that one line. I could have eventually found this with the standard trace utility, but would have ended up with gigabytes of trace logs and would have been poring over them for days. And assuming I could have found the error that way it would still have taken me a long time tracking down every place the file was opened and closed.

I'm sure I'm just scratching the surface of file callback functions, but so far they've turned out to be very helpful in tracking down issues while debugging.

Download the source

Reader Comments

Posted on Thursday, September 20, 2007 by Dave Harms

I've added a readme file from Marty to the source zip.

Dave

.....

Posted on Thursday, September 20, 2007 by JC Harris

Nice article. I wish you had provided at least one example of the callback function being used in a Clarion app to show how it's practical application. Or did I miss something?

.....

Posted on Monday, September 24, 2007 by Juan Herrera

Awesome!

I was thinking to do the same a couple of days ago and now it's solved! Thanks!!!!

.....

Posted on Tuesday, September 25, 2007 by Marty Honea

JC,

What kind of an example would you like? I'll be more than happy to put one together.

The class will display errors given by the file driver, along with documentation of when the error occured. With the example given, this can be seen if you try to scroll above the beginning or below the end of the file.

Marty

Add a comment

Clarion Magazine

Multiline Radio Buttons And Checkboxes

by Carl Barnes

Published 2007-09-18

Many times I'll have a RADIO or CHECK control with text longer than will fit on a single line. I would like the text to wrap over multiple lines exactly the way it does on a PROMPT control, but the Clarion radio and check controls do not support multiline text. The workaround I use is to put the first line of text in the control and the additional lines in a PROMPT control positioned below. This workaround functions just fine and may be visually acceptable to some users, but typically does not look perfect. The text does not line up exactly horizontally or vertically. The focus rectangle only wraps the first line of text which not only looks poor, but can be a little confusing. The final flaw is that when the user clicks on the workaround prompt text the control does not get selected. This lack of functionality can make user think the control is disabled or read-only.

Figure 1 shows the difference between the workaround and a true multiline control. In each case, the lower example shows the appearance with an XP manifest. As you can see, the workaround can have problems with text alignment.

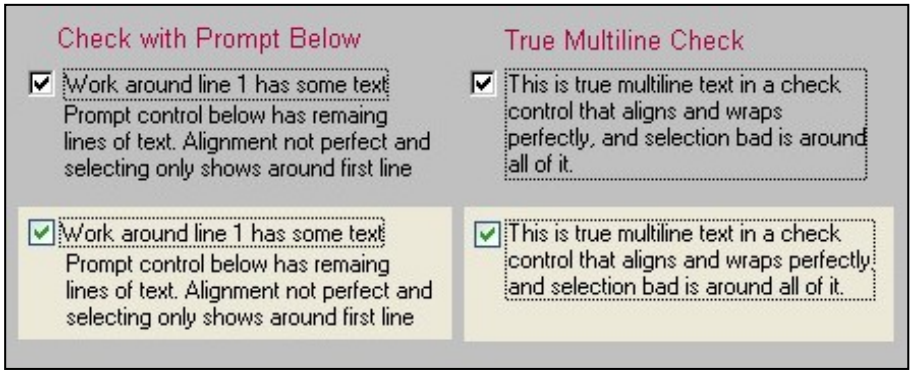


Figure 1. Workaround and true multiline controls

Clarion may not natively support multiline controls, but that doesn't mean it can't be done. In this article I'll show you how to easily add multiline button, radio, and check controls to your applications by changing one style bit via the API.

Start with the button

Most Clarion controls are based on some form of standard Windows control. The Clarion BUTTON, CHECK and RADIO controls are all based on the Windows Button control class. (Yes, a check and radio are special forms of a button.) Looking up the styles available for the Button class on MSDN I found BS_MULTILINE (0x00002000), with the description "Wraps the button text to multiple lines if the text string is too long to fit on a single line in the button rectangle".

You can modify control styles with the Window API function SetWindowLong(), protyped as follows:

```
SetWindowLong(Unsigned,SIGNED,LONG),LONG,|
  PASCAL,DLL(1),NAME('SetWindowLongA'),proc
```

The following code is all it takes to turn on the multiline style:

```
BS_MULTILINE    EQUATE(2000h)
GWL_STYLE       EQUATE(-16)
BtnStyle        LONG,AUTO
   CODE
   BtnStyle=GetWindowLong(?Check1{PROP:Handle},GWL_STYLE)
   BtnStyle=BOR(BtnStyle,BS_MULTILINE)
   SetWindowLong(?Check1{PROP:Handle}, GWL_STYLE, BtnStyle)
```

Why didn't TopSpeed or SoftVelocity implement multiline controls in Clarion? Probably because this style is part of the Win32 API. Clarion supported 16 bit compilation through version 5.5, and the Clarion 6 IDE remains 16-bit. The 16-bit issues would have made it difficult to implement the feature. And maybe the feature was never requested by a developer. Searching the web for BS_MULTILINE information I also found that Borland never implemented multiline in Delphi or C++ Builder, so those developers also have to add multiline support via the Windows API.

New Win32 Alignment Styles

Besides multiline buttons there were also button text alignment styles added in Win32 that the Clarion language and IDE do not accommodate. The vertical position of the text can be specified as top, center (default) or bottom. For a check or radio this also specifies the location of the check box or radio circle. The horizontal alignment of the text can be specified as left (default), center or right. I don't care for the default vertical alignment of centered, so I typically change it to Top. Figure 2 shows some of the alignment combinations.

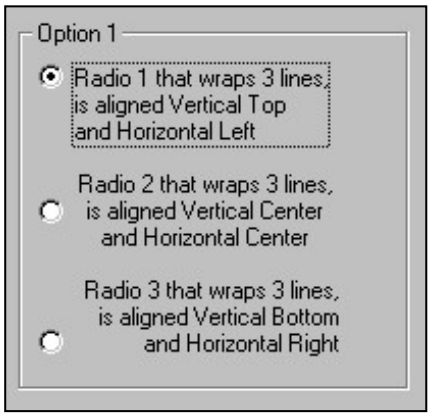


Figure 2. Left/top, center/center, and right/bottom alignments

The API also allows changing the placement of circle or box to be on the right side of the text with the style BS_RIGHTBUTTON (or you can use the alias BS_LEFTTEXT). Clarion does supports this style by adding the

LEFT attribute to the control. But Clarion does not allow you to specify right or centered text, so you end up with left aligned text as in Figure 3. I wish TopSpeed Clarion hadn't used the LEFT attribute to specify the location of the box/ circle; now the keywords LEFT, CENTER and RIGHT cannot be used to specify text alignment.

Figure 3. Clarion vs API right-side checkboxes

The complete list of Windows button style equates for setting these values are shown in the below table. Note that left, right, top and bottom are all separate bits. To turn on one you'll need to be sure the opposite is turned off, i.e. to turn on left you need to be sure right is turned off or you may end up with centered.

To turn off a bit you need to bitwise AND the complement. For example, to turn off Right the code is Style=BAND (Style,BXOR(-1,BS_RIGHT)).

Equate	Value	Description
GWL_STYLE	-16	Sets a new window style using SetWindowLong
BS_LEFT	0100h	Left-justifies the text in the button rectangle
BS_RIGHT	0200h	Right-justifies text in the button rectangle
BS_CENTER	0300h	Centers text horizontally in the button rectangle (Note Center = Left +Right)
BS_TOP	0400h	Places text at the top of the button rectangle
BS_BOTTOM	0800h	Places text at the bottom of the button rectangle
BS_VCENTER	0C00h	Places text in the middle (vertically) of the button rectangle (Note Center = Top+Bottom)
BS_RIGHTBUTTON	0020h	Positions the radio circle or check square on the right side of the button rectangle
BS_LEFTTEXT	0020h	Alias for BS_RIGHTBUTTON. Same as Clarion LEFT on RADIO or CHECK.
BS_PUSHLIKE	1000h	Makes a check box or radio look and act like a push button. The button looks raised when it isn't pushed or checked, and sunken when it is pushed or checked. Clarion CHECK or RADIO will render "Push Like" if they have an ICON or FLAT attribute.
BS_MULTILINE	2000h	Wraps the button text to multiple lines if the text string is too long to fit on a single line in the button rectangle
BS_FLAT	8000h	Specifies that the button is two-dimensional; it does not use the default shading to create a 3D image.

The BS_PUSHLIKE style renders a CHECK or RADIO as a "latch button" instead of a box or circle graphic. When unchecked the button appears in its normal up position, when checked it appears as latched down. This can be done in

Clarion code by adding an ICON() to a CHECK or RADIO. Specifying ICON(ICON:None) will render the latch effect as a text button without an icon showing. The FLAT attribute will also render as latch button, but the button will appear flat when unchecked (Figure 4).

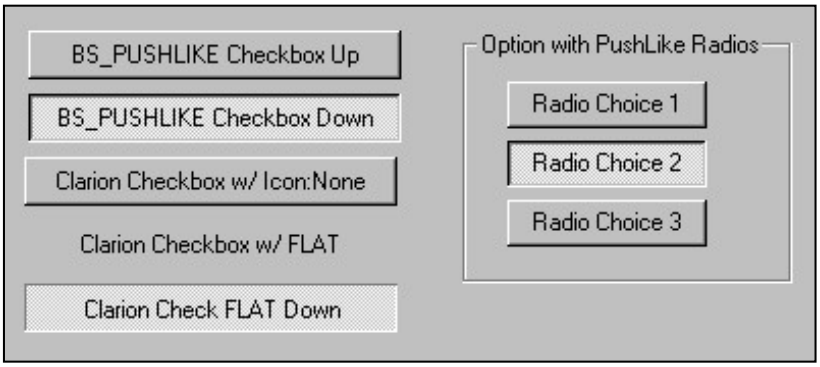


Figure 4.BS_Check/Radio's with BS_PushLike style

The API BS_Flat style is very different from the Clarion FLAT attribute. BS_Flat turns off 3D and renders a button, check or radio in a 2D effect that has an interesting retro look (Figure 5).

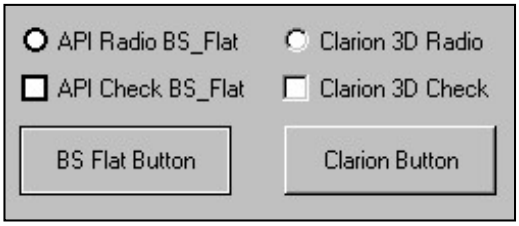


Figure 5.BS_Flat style

Multiline Text Buttons

Clarion supports multiple lines of text on a button as long as the button has an icon. For a multiline text only button with no icon at all use Icon:None:

```
BUTTON('With multiple<13,10>lines of text'),ICON(ICON:None)
```

In the Clarion C6 Window Formatter button icon drop list select "None" instead of "No Icon"; in C5.5. and prior you must add the attribute via the Window ellipsis [...] button on the procedures property. You can also skip the ICON attribute and apply the API BS_MULTILINE style via the API call. In my testing the results were identical so I would tend to use Icon:None and not have to write any extra code.

Specifying a font color using the FONT(,,COLOR:xxx) attribute for the text on the button will also cause Clarion to support multiline text on a button. If you don't want colored text you can use the equate COLOR:BtnText to use the Windows default color. COLOR:BtnText worked correctly at runtime but did not display correctly in the IDE. And finally, the FLAT attribute will also wrap the button text.

Manifest Issues

If you have an XP Manifest and the user's Windows appearance is set for the Classic look, then the horizontal alignment of center and right does not quite work the same. It appears the first line is done using the requested alignment, but the rest of the lines are always left -aligned under that. Figure 6 is a screen capture showing without and with an XP manifest in the Classic look.

Figure 6. Manifest alignment differences with the Classic look

Multiline in the Clarion IDE Window Designer

The Clarion Window Designer does not support any of these API multiline text or alignment features at design time or in preview mode, so what you see in the IDE is not what you'll get at runtime. You'll just see one line of text centered in the height of the control and additional lines will be truncated. Trying to size the control will require running the program to verify it displays as desired.

One way to get a design time view is to use a PROMPT control as a design-time tool, but not as a runtime element on the form. You put the full text into a PROMPT control next to the CHECK / RADIO, then at runtime transfer the text and resize the control; once CHECK / RADIO control is set up you disable and hide the PROMPT.

The PROMPT supports specifying the Alt key with an ampersand and this hot key will also transfer to the CHECK / RADIO control. The PROMPT also supports embedding manual line breaks with a <13,10> in the text, and this will work correctly with multiline button controls.

Figure 7 is a screen shot from the Window Designer showing both methods. I selected all the controls so you can see that on the left the text is a separate prompt control next to the check control.

Figure 7. The Window Designer showing both approaches to multiline controls

In most of my work the text fits into two lines so I won't bother with the separate prompt. I temporarily add a prompt to get an idea of the size required to fit and wrap the text. Then I'll put the text into the check/radio, resize it using the prompt, and delete the prompt. I size the control wider then the first line of text and insert a <13,10> where I want the break to occur.

A Function to Make it Easy

Included in the download example is a function named SetBS_Multiline that makes it easy single call to implement all of the styles discussed above. The prototype for the function is as follows:

```
SetBS_MULTILINE procedure(  |
    LONG ButtonFEQ,       | ! FEQ BUTTON, CHECK or RADIO
    LONG PromptFEQ=0,     | ! FEQ PROMPT for text and size
    BYTE xLeftCntrRght=0, | ! 1=Left 2=Center 3=Right
    BYTE yTopCntrBttm=1,  | ! 1=Top  2=Center 3=Bottom
    BYTE bRghtSideBtn=0,  | ! 1=Button on Right
    BYTE bMultiLine=1)    | ! 1=Multiline
```

This function is designed to optionally support your window design having a PROMPT containing the multiline text next to the CHECK / RADIO. The prompt must have the desired width and be aligned to where the text typically starts. The CHECK / RADIO will be resized to the height and width of the PROMPT, and it will allow width for the box/radio button as the prompt x position minus the control x position. Buttons are not resized by the function.

Figure 8 shows a number of controls in the Window designer (with all controls selected).

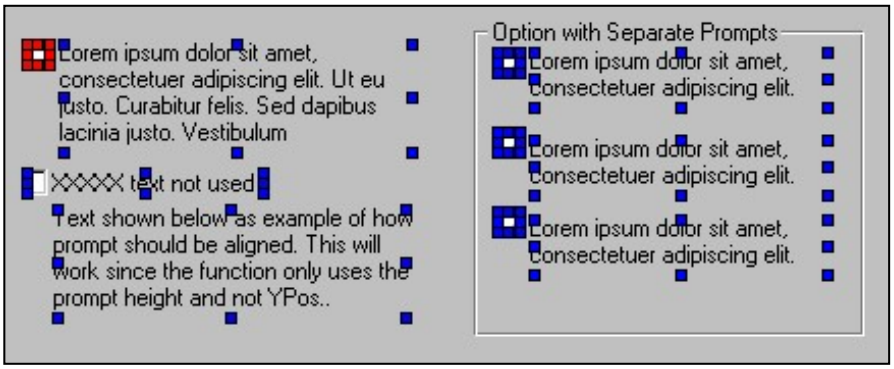


Figure 8. Various controls in the Window Designer

SetBS_Multiline is designed to be called after OPEN(Window) and before the ACCEPT. If you call it inside the ACCEPT loop the controls will not repaint immediately. Issuing a ?Check{PROP:Right}='1' should force the RTL to repaint.

The source code for this function is fairly simple, as follows:

```
SetBS_MULTILINE procedure(  |
    LONG ButtonFEQ,       | ! FEQ BUTTON, CHECK or RADIO
    LONG PromptFEQ=0,     | ! FEQ PROMPT for text and size
    BYTE xLeftCntrRght=0, | ! 1=Left 2=Center 3=Right
    BYTE yTopCntrBttm=1,  | ! 1=Top  2=Center 3=Bottom
    BYTE bRghtSideBtn=0,  | ! 1=Button on Right
    BYTE bMultiLine=1)    | ! 1=Multiline
```

```
WindStyle      LONG,AUTO
Style_ON       LONG(0)
Style_OFF      LONG(0)
CtrlHWnd       UNSIGNED,AUTO
ControlType    LONG,AUTO
Prmt_X         LONG,AUTO   !Prompt Position
Prmt_Y         LONG,AUTO
Prmt_W         LONG,AUTO
Prmt_H         LONG,AUTO
Btn_X          LONG,AUTO   !Button Control's X
Btn_W          LONG,AUTO   !Control's X
  CODE
  CtrlHWnd = ButtonFEQ{prop:handle}
  WindStyle=GetWindowLong(CtrlHWnd,  GWL_STYLE)
  IF bMultiline
    Style_ON=BOR(Style_ON,BS_MULTILINE)
  end
  IF INRANGE(yTopCntrBttm,1,3)
    Style_OFF=BOR(Style_OFF,BS_VCENTER)
    Style_ON=BOR(Style_ON, |
          CHOOSE(yTopCntrBttm, BS_TOP,BS_VCENTER,BS_BOTTOM))
  END
  IF INRANGE(xLeftCntrRght,1,3)
    Style_OFF=BOR(Style_OFF,BS_CENTER)
    Style_ON=BOR(Style_ON, |
           CHOOSE(xLeftCntrRght, BS_LEFT, BS_CENTER, BS_RIGHT))
  END
  IF bRghtSideBtn
    Style_ON=BOR(Style_ON,BS_RIGHTBUTTON)
  END
  WindStyle=BAND(WindStyle,BXOR(-1,Style_OFF))
  WindStyle=BOR(WindStyle,Style_ON)
  SetWindowLong(CtrlHWnd, GWL_STYLE, WindStyle)
  IF PromptFEQ
    CASE (ButtonFEQ{prop:Type})
    OF Create:Radio OROF Create:Check
      GETPOSITION(PromptFEQ,Prmt_X,,Prmt_W,Prmt_H)
      GETPOSITION(ButtonFEQ,Btn_X,,Btn_W)
      IF Btn_X < Prmt_X          !Button Left of Prompt?
```

```
      Prmt_W += (Prmt_X-Btn_X)
    ELSE                    !Button Right of Prompt?
      Prmt_W =  Btn_X + Btn_W - Prmt_X
      Btn_X  =  Prmt_X
    END
    SETPOSITION(ButtonFEQ,Btn_X,,Prmt_W,Prmt_H)
  OF Create:Button
    !Buttons assumed to be the sized right
  END
  ButtonFEQ{prop:Text}=PromptFEQ{prop:Text}
  HIDE(PromptFEQ)
  DISABLE(PromptFEQ)
END
RETURN WindStyle
```

Double Click Radio

A somewhat common dialog offers the user a single choice implemented as an option with radio buttons, and a button to execute the choice. Figure 9 shows an example of such a window.

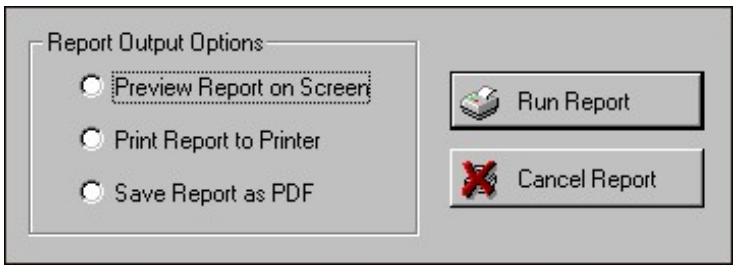


Figure 9. Single Option Dialog

After selecting the radio I find it annoying to have to mouse to the right side and click the button. What I want to do is double click on the radio button and have the logical choice, pushing the run button (default button), done for me. This behavior is easy to program in Clarion by alerting the MouseLeft2 key on the OPTION and processing the EVENT:AlertKey in each RADIO. Note the detail that only the radio buttons get the alert events. An example of double-click radios is included in the download.

Further Reading

The book *Win32 Programming* by Brent Rector and Joseph Newcomer has excellent coverage on programming dialogs using the standard Windows controls and the Windows GDI. It's a 1500 page book and deals with most of the Win32 API. One nice thing for Clarion programmers is it highlights differences between Win16 and Win32 in prominent tip boxes. It was in this book I found out BS_MULTILINE was not available under Win16. Most Win16 information is gone from MSDN.

The CD included with the book contains many useful utility programs. Some of the coolest are the "Explorers" which let

you easily configure the many options available for the Windows programming concepts explained in the book. The Control Explorer will let you test all of the Windows standard controls and configure any option as well as send and view messages. This utility is useful for finding other features supported by the API but not by Clarion. Figure 10 is a screen shot of Control Explorer in which I have specified a checkbox as multiline:

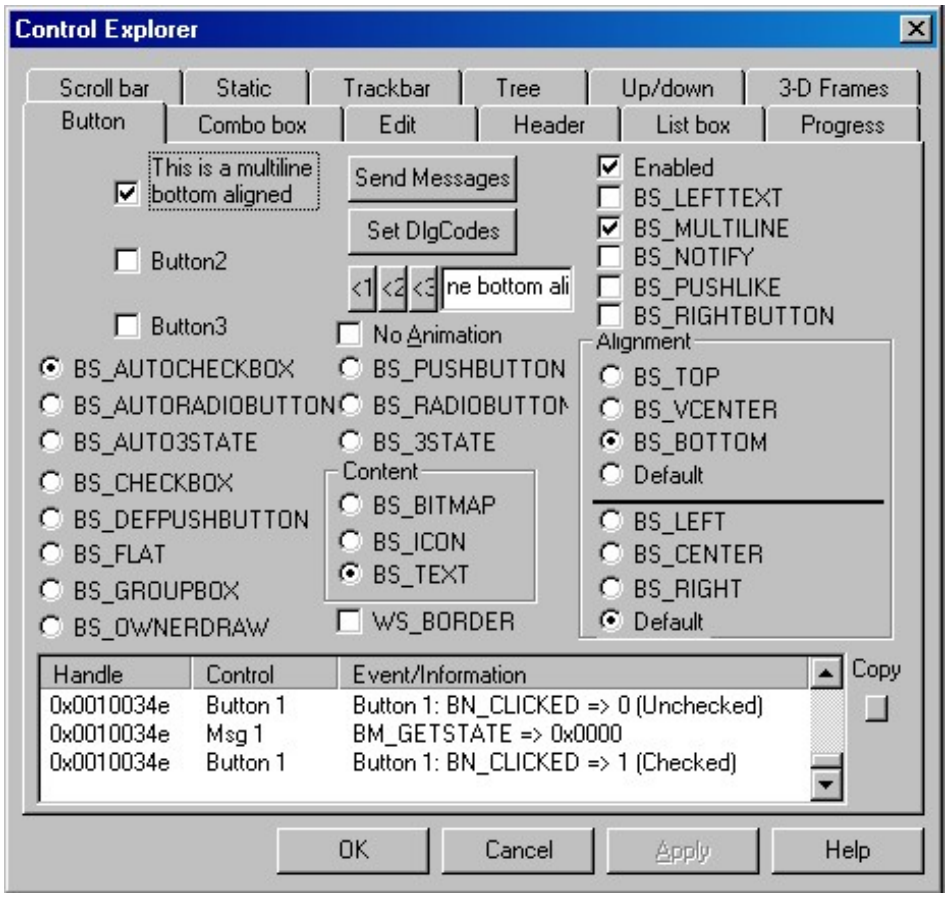


Figure 10. The Control Explorer

Clarion 7

The Clarion 7 Beta build 2430 does not implement multiline or any of the other styles discussed in this article. The Structure Designer will correctly display multiline check/radio text if it contains <13,10> line breaks, but the RTL does not render them as multiline at runtime. All code supplied with this article has been tested and works correctly under 7.2430, as well as Clarion 6, 5.5 and 5b.

Summary

That's about everything I know about button styles supported by the Windows API. There are three-state check boxes (checked, unchecked, and partly checked), but that's a story for another time. You may find it interesting to explore MSDN for other Window control styles not supported by Clarion.

Be sure to download and try the example project. There are examples of every button style I discussed in this article, and a few extended styles that were notI didn't covered. The function it contains is all you'll need to easily implement these new styles. This function could be built into an ABC compliant class or utility function DLL to make it easy to reuse in all

your applications.

Resources

- MSDN Button Reference:
- MSDN Button Styles

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.

Reader Comments

Add a comment

Clarion Magazine

Waiting For Files With Clarion Threads

by Alan Telford

Published 2007-09-12

By nature I am a *moderately* conservative person (did you ever notice that conservative people will always describe themselves as *moderate*, and not *ultra*-conservative). I don't use new technology just for the sake of it. When Clarion 6.0 first came out it had the "huge advantage" of preemptive operating system threads where tasks could actually run in the background. Well, so what! How's that going to help me? I didn't particularly know or care about background threads. Then about mid-2006 the nature of my work took a turn.

Previously I had been involved primarily in writing desktop applications to be installed at single stand-alone sites. This was bread and butter for me. I was comfortable doing this work – right at home. But our target market was rapidly shrinking and we had been unable to make any significant new sales for some months. This led our company to switch to a new focus – a background program which silently polled a cash register and transmitted the data to a central web server.

The site component had to run silently without any user interaction, collect data, and transmit to a central server. The server had to receive, merge and then report on the data. And of course it had to operate 24x7, non-stop, rain or shine, wind or hail, night or day, whether I was awake or asleep. And it had to be fast. Suddenly I was out of my depth and no longer the confident Clarion programmer who'd done it all before.

In a daze, overwhelmed by these new ill-defined requirements, I looked for something nice and simple that I could tackle first. I decided that both site and server components would need a mechanism for watching a folder and responding as soon a new file appeared. The site should send all files from this folder to the central server. When they arrive the server should instantly respond and merge them into the database. Hmmm. I remembered a ClarionMag article that discussed Windows API calls both for discovering new files in a directory and for running a thread for that specific task.

I read both parts of Jim Kane's excellent article. I did feel demoralized that I was seven years behind the times (the article was written back in 2000, before Clarion 6 and its preemptive threading capability) but I was encouraged that at least I had remembered the article and wasn't wasting all my time re-inventing the wheel.

Previously I had been using the DIRECTORY() function inside the EVENT:Timer embed point to check for new files. Jim Kane says:

> A window timer will always impact performance and will not run under some circumstances. If it is important to detect the file almost immediately and not affect performance there are better ways with the API than a timer.

I downloaded the source code and attempted to run the *changenotif.exe* only to be given the "c5runx.dll not found" message. No problem. I opened the PRJ file in Clarion 6.3, recompiled and ran the program. It still worked even in the year 2007. Alright! However it was it was written for Clarion 5 and therefore had to use API threads. Now that Clarion 6 was here I should be able to use Clarion threads directly. I decided my first learning challenge was to learn about threads by updating the code to use the new Clarion 6 features.

As I studied the code I discovered the complexity of the API had been hidden very nicely in Jim's class. The Clarion program simply had to declare an object and define which custom event number should be posted back to the window when a file change was received. At the Event:OpenWindow embed point the object was initialized, and on the custom event

embed the user should take whatever action they required.

My job is not to describe what Jim has already done, as this is already well handled in his earlier articles. Instead I just want to highlight the changes required to make Jim's code work with Clarion 6's preemptive threading.

Starting threads and passing parameters

In Clarion 5 you use createthread operating system function start a new thread. You must also remember to clean-up and close the handle. The thread can receive a single long parameter, so you store information in a local group and pass the address to the new thread.

```
SELF.Threadhandle =
  createthread(0,256,address(waitproc), |
  address(SELF.waitstruct), 1, threadid)
! when finished make sure you clean up
Closehandle(SELF.ThreadHandle)
```

In Clarion 6 you simply use the start command to start a new thread. No cleanup is required. The start command can receive three string parameters. I decided for convenience to pass the actual class object into the procedure, by address.

```
SELF.waitThread = start(WaitProc,,address(self))
```

Accessing the passed parameters

In Clarion 5 you declare a group structure of the same type you declared locally, and you use the Windows memcpy function to copy the contents of the original group into the local group.

```
WaitProc PROCEDURE(long lpwaitstruct)
loc:WaitStruct LIKE(WaitStructGroupType)
infinite       EQUATE(-1)
Wait_object_0  EQUATE(0)



  CODE
  memcpy(ADDRESS(loc:waitstruct),lpwaitstruct,SIZE(WaitStructGroupType))
```

In Clarion 6 you take advantage of Clarion's automatic type conversion and you assign the numeric value of the passed string (the group's address) to the local reference:

```
WaitProc  PROCEDURE(STRING pClass)
cWait          &MTChgNotifyClass
  CODE
  cWait &= pClass + 0 ! get a reference to the class
```

Signaling the Clarion window

In Clarion 5 a *lot* of steps are required to post an event from the thread back to the Clarion window. Firstly, when initializing the object the window must be subclassed and the RegisterWindowMessage function used before inter-thread communications can be used. Then SendMessage is used to send the registered message back to the Clarion window. The sub-classed procedure must receive this message (Clarion 5 cannot) and then post a Clarion event to the window. No parameters are allowed with the post command, just the number of the event.

```
! cnotifClType.Init() method, register a unique
! message, and subclass the window
SELF.WaitStruct.wMsg = RegisterWindowMessage(pTargetdir)
savedproc=0{prop:wndproc}
0{prop:wndproc}=address(subclassproc)
...
! WaitProc PROCEDURE() send the unique message to
! the Clarion window
Sendmessage(loc:waitstruct.hwnd,loc:waitstruct.wmsg,0,0)
...
! SubClassProc PROCEDURE() finally post event to clarion window
POST(SaveGrp.CWEvent,,SaveGrp.CWThread)
```

In Clarion 6 posting an event from the thread is far simpler. You can use the new Clarion 6 notify command directly. Even better, it allows not just a notify code but also an additional parameter.

```
NOTIFY(cWait.cwnotifycode, cWait.cwthread, cWait.cwnotifyParam)
```

I didn't use the additional parameter until I started watching multiple directories. One class object must be instantiated to watch each directory, and the object then signals back to the main thread when action is required. But how does the main thread recognize which object sent the event? In Clarion 5 a different event number is required to associate an object with an event which object it belongs to. However in Clarion 6 the same notifycode can be used for all cases, while the optional notifyparameter can identify which object originated the event.

This covers the differences between Jim's code and my C6 version. But how is the object actually used inside your program? Very similar code is used to declare the object in local data, initialize the object inside event:openwindow, and clean-up the object at event:closewindow. The main change is in receiving the signal that the file-change has occurred.

Receiving file change messages

In Clarion 5 you must declare a custom event number (e.g. in local data). You then check for this event inside the accept loop and call your event handling code:

```
event:FileChanged equate(888H)
  ...
 case event()
   of event:FileChanged
     Message('FileChanged') !put your desired action here
```

In Clarion 6 the notify() command triggers the built in event:notify event at which point you must use the companion notification() command to retrieve the notify code (and the optional parameter). So rather than declaring a

custom event, you declare a custom notify code:

```
Case event()
Of event:notify
  if notification(notifyCode, , notifyParam)
    if notifyCode = eMyNotifyCode !notifyParam=0
      !processing here of the .\trigger folder
    end
  end
end
```

Summary

I hope this comparison has served to motivate you to try out the new Clarion threading features. Sure it was possible to use API threads in Clarion 5, but I suspect it would have been too much work for many Clarion programmers. However in Clarion 6 we really no longer have an excuse.

Download the source

Alan Telford has been programming in Clarion since 1994. He is the Chief Software Developer at Maxtel Software Ltd, a New Zealand software company specializing in writing back office computer solutions for McDonald's Family Restaurants and other similar markets.

Reader Comments

Add a comment

Clarion Magazine

Upcoming articles on C7, Clarion.NET, and more

Published 2007-09-07

Blogs on this site

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

Any time I come back from vacation I expect a busy week or two, but this time around was a bit busier than most. The ClarionMag web server, which had been exhibiting some small signs of instability, began crashing several times a day while I was still three days drive from home. Switching to a new server at a time of crisis is always a hassle, if for no other reason than it takes time for the DNS changes to propagate. With the main server limping along and needing frequent reboots, I got the backup server ready. At the last moment my ISP found a motherboard problem and offered to swap the RAID array into a new server. Once again everything is (I trust) rock solid.

I have a full slate of articles lined up for September, but due to time I spent on server issues they won't start showing up until next week. Topics include (but are not limited to) compiling C6 applications in C7, more on PostgreSQL, and how language changes in Clarion.NET will affect your development (there's a whole lot of very interesting Clarion.NET info in the C7 help file).

Meanwhile CSP members are digging into the first beta release, and we're all hoping to get the data diagrammer shortly. I'll have a report on that new Enterprise Edition tool as soon as I can - meanwhile you may want to read Bob Zaunere's blog post on the subject and watch the movie.

Clarion Magazine

The Clarion 7 Beta Release

by Dave Harms

Published 2007-09-12

Last week SoftVelocity released the first C7 beta to participants in the Core Subscription Program (CSP). As noted in the SoftVelocity store, the CSP, which is normally a one year program covering major and minor upgrades, also includes the upgrade to Clarion 7.0, independent of when that product is released.

This first C7 beta release is much like the alpha release ClarionMag reported on in July, with some fixes and minor changes. Key features included in the beta are:

- The new IDE, which is written in .NET and is part SoftVelocity's code, and part SharpDevelop, and open source IDE for which SoftVelocity has a commercial source license (and no, that's not a contradiction in terms, as the makers of SharpDevelop are free to release their code under multiple licenses).
- The Clarion compiler
- The structure designers (window and report)
- The dictionary editor (with some limits on functionality)
- The database browser

The two main features still to be completed are:

- The new Data Diagram tool
- The AppGen

While a lot of developers are looking forward to the new data diagrammer (you can see a video here), the item we're all waiting on is most definitely the AppGen. And that's not available yet.

So what you can do with the beta?

Plenty.

Most importantly you can import C6 applications as source code and compile them with the C7 compiler and runtime library. That matters because C7 includes the following features:

- Support for ClearType/unicode
- Eye candy - that is, support for visual styles and tab improvements

ClearType is particularly important if your users are on LCD monitors and have ClearType turned on, in which case C6 applications may not display text properly.

Figure 1 shows a variety of fonts in entry fields on C6 with ClearType enabled. While MS Sans Serif still works, most other fonts show fading of strokes particulary on narrow characters.

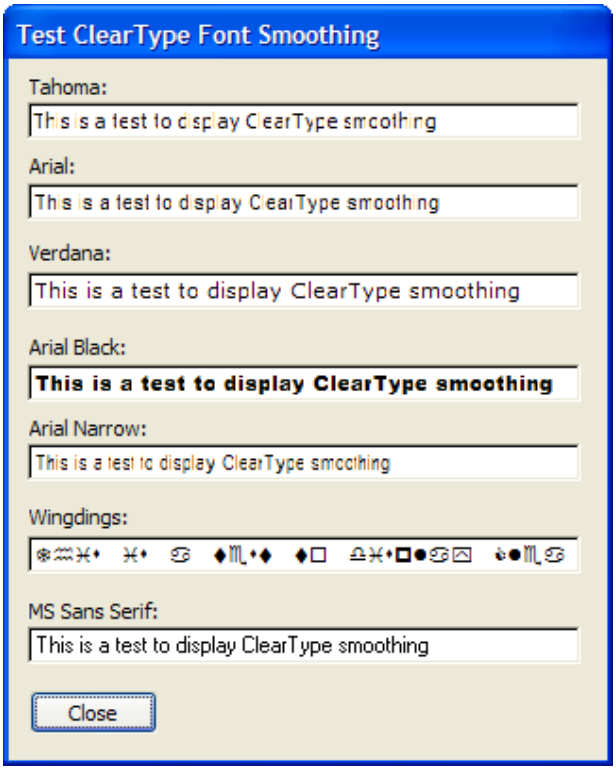


Figure 1. ClearType and C6 apps

Figure 2 shows the same fonts but on a C7-compiled application. No changes were made to the application itself - it was simply recompiled in C7.

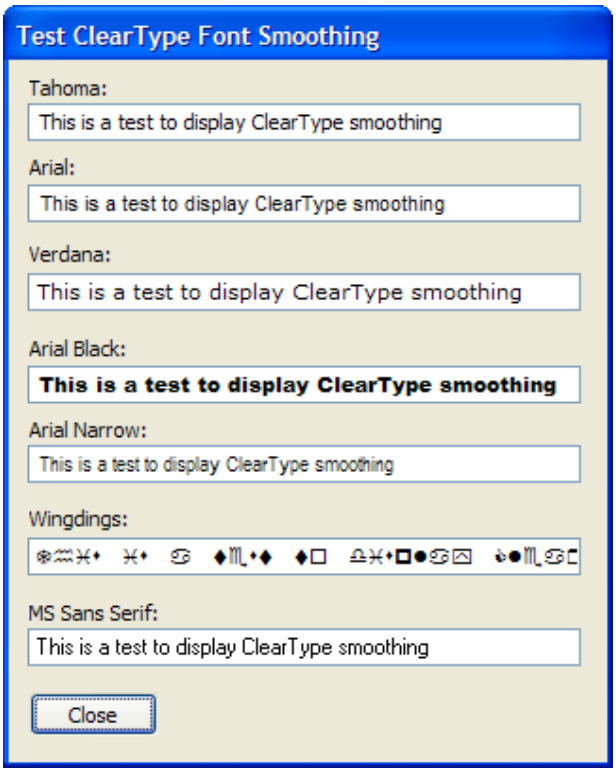


Figure 2. ClearType and C7 apps

The C7 runtime also has support for XP styles, and the C7 installer includes a template patch to add the corresponding Clarion code to the C6 templates. For more on the new visual styles see the article First Look: Clarion 7 Alpha, Part 2.

The new IDE

I'll cover the process of compiling a C6 application in C7 in a moment. But first I'd like to emphasize (not for the first or last time) that C7 presents an IDE that's radically different from previous Clarion IDEs. Those were closed systems, tailored to Clarion's specific needs. With C7/Clarion.NET (both products share the same IDE) SoftVelocity has departed from that tradition. Yes, when C7 is finished you'll still have all that Clarion-specific functionality *including the AppGen*. But you'll have a whole lot more, thanks to the SharpDevelop code base.

SharpDevelop is an open source IDE, initially targeted at C# developers. As such it looks and works a whole lot like Visual Studio, but as an open source project it's also highly extensible via add-ins. In fact, much of what SharpDevelop does, it does via add-ins, and its configuration information is stored XML files.

> **NOTE:** In earlier alpha versions the Clarion IDE included support for languages such as C#, ASP.NET, Boo, etc., but these have been removed, no doubt to avoid confusion in what is already a complex development environment. It appears at least possible you could add these back in if you wanted, although ideally you'd compile the appropriate plugin with the same build of SharpDevelop as SV uses in its code base. My guess is some of this language support, say C#, will make its way back into the shipping Clarion IDE at some point.

It's an oversimplification to say that the Clarion bits simply plug in to the SharpDevelop IDE's receptacles; as Bob Z can tell you the wiring is a whole lot more complex than that. The new IDE is more like a synthesis of Clarion and SharpDevelop. And with all those components, some from SV's bag of tricks, some part of SharpDevelop, it's a very big IDE - overwhelmingly so at first.

Here are just some of the features you'll find in C7 right now:

- code folding
- code completion
- AddIn Manager
- Component Inspector
- support for multiple languages
- real search/replace and undo in the text editor
- new structure designers
- property toolbox
- fully customizable source color highlighting (templates too!)
- error logs
- console output
- bookmarks
- multiple configurable layouts
- task list
- source code formatting
- regular expression toolkit
- support for multiple Clarion versions
- MSBuild-compatible solution files
- a bazillion configuration options

In other words, there's a lot to get up to speed on in the absence of the AppGen (and I'm sure I've missed a few important

items in the list).

Compiling C6 apps in C7

Until the C7 AppGen arrives, you can generate your code in C6 and compile in C7; all you need to do is export the project data from your C6 application, then open that PRJ file in C7 and compile.

Project data is made up of a list of source files and compiler/linker instructions, something like the following (taken from the Examples, HowTo-ABC, Browses app):

```
-- Generator
#noedit
#system win32
#model clarion dll
#pragma debug(vid=>full)
#pragma optimize(cpu=>386)
#pragma define(_ABCDllMode_=>0)
#pragma define(_ABCLinkMode_=>1)
#pragma define(_GRAPHDllMode_=>0)
#pragma define(_GRAPHLinkMode_=>1)
#pragma define(_svDllMode_=>0)
#pragma define(_svLinkMode_=>1)
#pragma define(_MSSQLDateTime_=>1)
#pragma define(_ORACLEDateTime_=>0)
#pragma link_option(icon=>"LOG1.ICO")
#compile "EFOCUS.CLW" -- GENERATED
#compile "ABCBRBC0.CLW" -- GENERATED
#compile "ABCBRBC.CLW" -- GENERATED
#compile "abcbrws.clw" -- GENERATED
#compile "abcbr001.clw" -- GENERATED
#compile "abcbr002.clw" -- GENERATED
#compile "abcbr003.clw" -- GENERATED
#compile "abcbr004.clw" -- GENERATED
#compile "abcbr005.clw" -- GENERATED
-- a bunch more #compile --
#compile "abcbr065.clw" -- GENERATED
#pragma link("C%V%DOS%X%%L%.LIB")
#pragma link("C%V%TPS%X%%L%.LIB") -- GENERATED
#pragma link("C%V%ASC%X%%L%.LIB") -- GENERATED
#pragma link("QkQBE.ico") -- GENERATED
#pragma link("QkLoad.ico") -- GENERATED
#pragma link("CheckOn.ICO") -- GENERATED
#pragma link("CheckOff.ICO") -- GENERATED
#pragma link("CheckOnDim.ICO") -- GENERATED
```

```
#pragma link("CheckOffDim.ICO") -- GENERATED
#pragma link("uncheck.ico") -- GENERATED
#pragma link("Star1.ico") -- GENERATED
#pragma link("12.ico") -- GENERATED
#pragma link("2.ico") -- GENERATED
#pragma link("5.ico") -- GENERATED
#pragma link("47.ico") -- GENERATED
#pragma link("4.ico") -- GENERATED
#pragma link("folder.ico") -- GENERATED
#pragma link("file.ico") -- GENERATED
#pragma link("invoice.ico") -- GENERATED
#pragma link("Check.ico") -- GENERATED
#link "abcbrws.EXE"
```

Project data is stored in any of three ways: within the binary .app file, in a text .prj file, or in a text .pr file. Both .prj and .pr files are typically used in hand-coded applications only; the only difference is that Clarion will open a .pr file in the text editor, and a .prj file in the project editor.

The easiest way to get project data out of an APP file is to open the APP and choose File|Export Project File from the main menu. Back before this option was available you basically had two choices: export a TXA and look for the [PROJECT] section, or if you were feeling particularly brave you could open the .app file in notepad (after making a backup, naturally) and search for the project data. Exporting via the main menu is so much easier, not to mention safer.

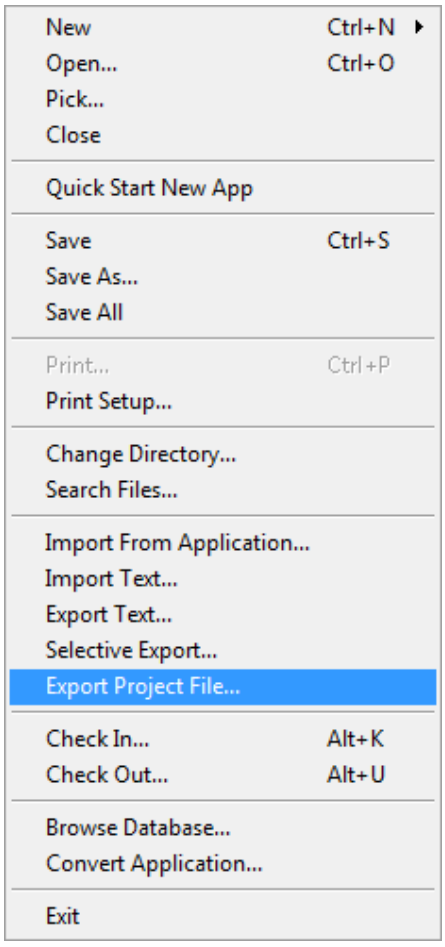


Figure 3. Exporting project data

If you look in the application's directory you'll see a file named *appname*.prj which contains all of the project data. In Clarion 7 choose File|Open|Project/Solution from the main menu. Navigate to the C6 application's directory; the dialog will look something like Figure 4.

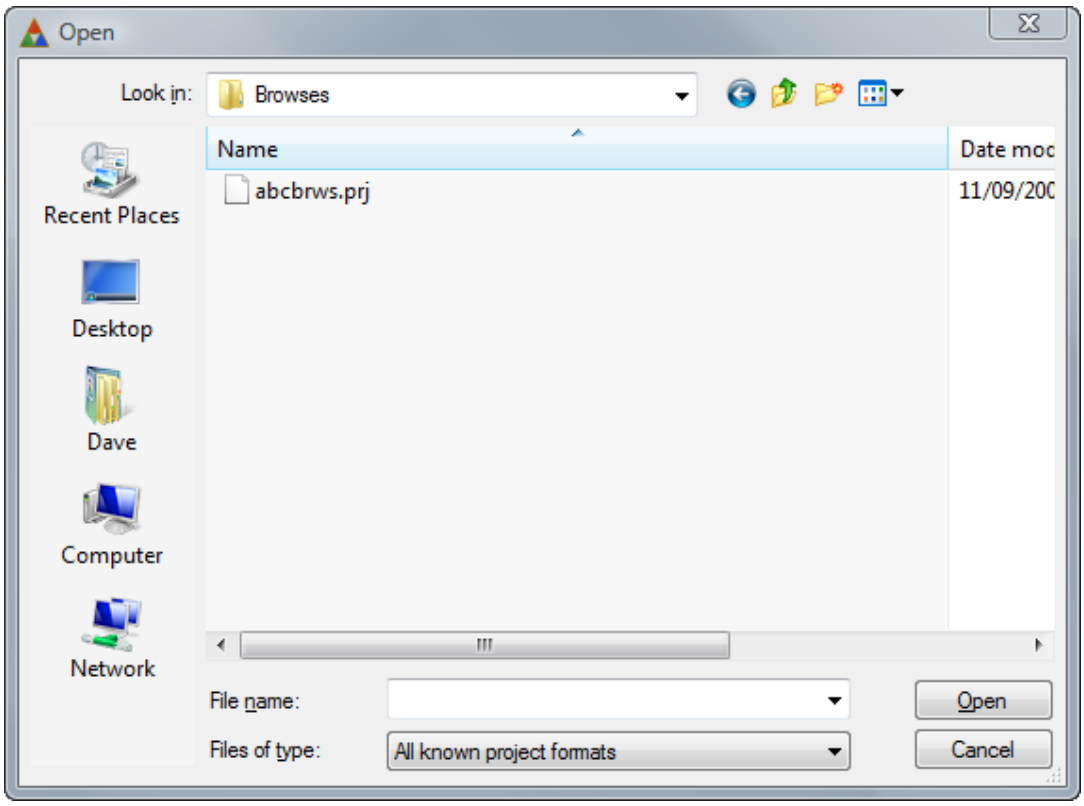


Figure 4. Opening the project in C7

C7 beta recognizes .pr and .prj files, as well as two new types: .cwproj and .sln, which are Clarion 7 projects and solutions, respectively. A .cwproj file contains all the project data for one application; a .sln file contains information about one or more applications. Once the AppGen ships this list will, I'm sure, also include .app files.

After C7 loads your exported .prj file it creates both a .cwproj and a .sln file for the application. These show up in the IDE's project explorer as shown in Figure 5.

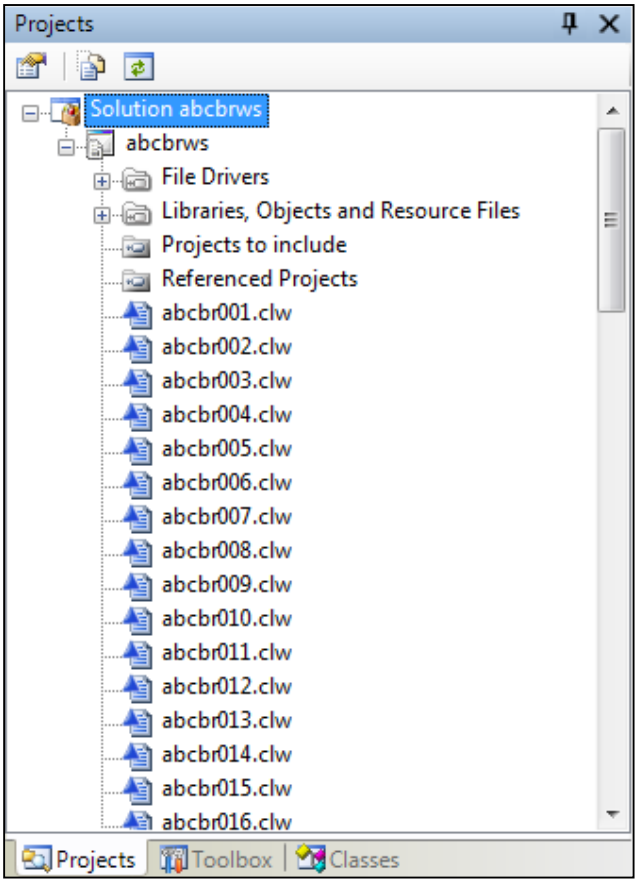


Figure 5. The project explorer

The first item in the Projects list isn't actually the project, it's the solution; the second item, in this case abcbrws (originally from the C6 abcbrws.app) is the project. You can build either the project or the solution at this point and achieve the same result, since there is only one project in the solution. But right-click on the solution and you'll see the context menu in Figure 6.

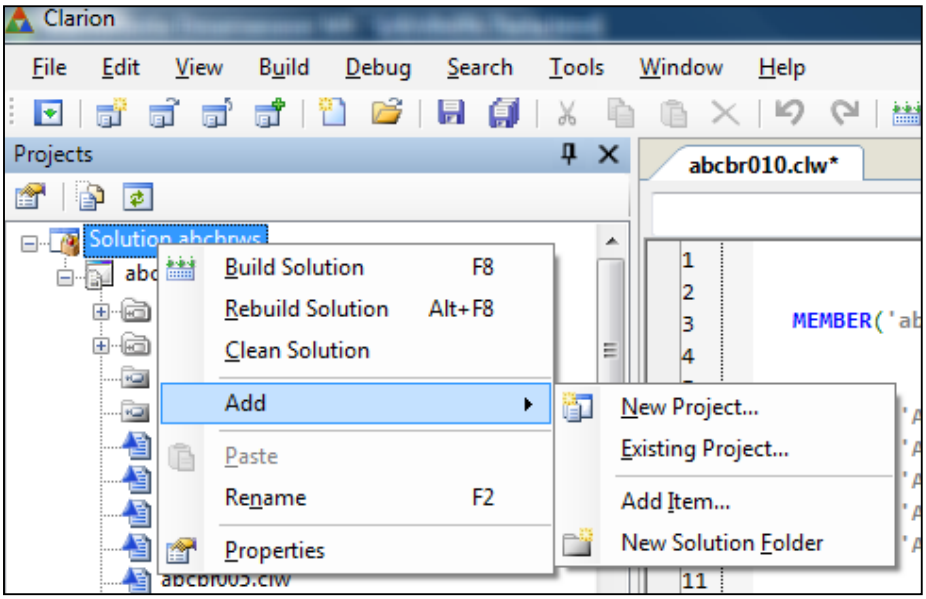


Figure 6. The solution context menu

As Figure 6 shows you can add more than one project to a solution so you can compile multiple applications using a

single command. It's up to you whether you build an individual project or the entire solution since the project also has a Build command on its context menu.

Judging by the responses of alpha and beta testers, compling C6 applications in C7 is pretty straightforward. If you're using any pre-compiled third party tools, however, you'll need to get C7 versions of those libraries.

Do keep in mind that if you make changes to the APP involving new or renamed source files, you'll need to make corresponding changes to the C7 project, or again export and import the C6 project data.

Resources

- Check out the C7 alpha/beta subtopic for previous Clarion Magazine articles on this subject
- The e-book Dissecting a C# Application: Inside SharpDevelop is written by members of the SharpDevelop team and provides some good insight on how the IDE is structured. It was formerly available from Apress, but doesn't appear to be on that site anymore. You can download it here.
- CSP for Clarion Professional Edition is currently listed at $400; Enterprise is $800. If you're wondering how to get in on the beta, just get yourself a CSP membership.

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

Reader Comments

Add a comment