

Clarion Magazine

Clarion News

- » » [Lodestar Software Adopts Subscription Model](#)
- » » [Easing into Clarion 7 - Part Two - Dictionaries](#)
- » » [Easing into C7 - Part One - Creating Something New](#)
- » » [C7 Video Part 3](#)
- » » [C7 Video Part 2](#)
- » » [UK User Group Web Site Revamped](#)
- » » [FullRecord 2.20](#)
- » » [CoolFrames 1.19](#)
- » » [Clarion 7 Video Part 1](#)
- » » [Clarion SQL Kickstart](#)
- » » [Clarion Folklore Podcast #3](#)
- » » [Product Scope 7 Video Tutorial](#)
- » » [Noyantis CommandBars 1.13](#)
- » » [Noyantis TaskPanel 1.04](#)
- » » [Duplicate Symbols Blog Entry](#)
- » » [Noyantis CommandBars 1.12 Beta](#)
- » » [iQ-XML For Clarion 6.3.9059](#)
- » » [Icetips Utilities 1.1.2324](#)
- » » [SetupBuilder 6.9 Build 2415](#)
- » » [Noyantis ShortcutBar 1.11](#)
- » » [Noyantis CalendarPro 1.11](#)
- » » [WindowToWinForm Converter](#)
- » » [Inback 1.8 Codes Sent](#)
- » » [DMC 1.5.0.5](#)
- » » [ABCFree Templates and Tools Updated](#)
- » » [EasyCOM2INC 2.09](#)
- » » [CNetCal 1.0](#)
- » » [BoTran2 1.0](#)
- » » [ClarioNET 6.0302](#)
- » » [ABCFree Templates And Tools Updated](#)
- » » [List & Label 14](#)
- » » [GTL6.41](#)
- » » [BFlat 1.2](#)
- » » [DMC Gold 1.5.0.4](#)
- » » [Free Mallorca Collection Subset](#)
- » » [Huenuleufu Templates For C7](#)
- » » [Freeware Locus Templates C7 Compatible](#)
- » » [Noyantis RSS Feed](#)
- » » [Noyantis CalendarPro 1.10 Beta](#)

[\[More news\]](#)

- » » [Clarion.NET FAQ](#)
- » » [Clarion# Language Comparison](#)
- » » [Code Snippets In The New IDE](#)
- » » [Understanding Clarion# Events](#)

Save up to **50% off ebooks.**
Subscription has its rewards.



Latest Subscriber Content

Edit-In-Place DropLists

Using the DropList class with ABC's Edit-In-Place (EIP) is reasonably trivial - so long as you've done it before. The problem is there's always a first time for doing it. In this article CapeSoft's Geoff Thomson walks through the creation of a simple EIP drop list.

Posted Friday, November 28, 2008

#FINDing Values In Prefixed Structures

If your template needs to #FIND prefixed local or global data you may have a problem (at least prior to C7, and maybe C6 9059). Lee White's template code offers a fix for #FIND.

Posted Friday, November 28, 2008

Code Snippets In The New IDE

The new Clarion IDE contains a nifty feature called code snippets. These are a one-way template-like tool that let you insert complex blocks of code with just a couple of clicks/keystrokes.

Posted Friday, November 28, 2008

Understanding Clarion# Events

Event handling in Clarion# (as in other .NET languages) is a potentially confusing subject. But as Randy Rogers and Dave Harms demonstrate, an EVENT in Clarion# is really just a special kind of delegate.

Posted Wednesday, November 26, 2008

Understanding Clarion# Delegates

Clarion# introduces a number of important new concepts, including delegates and events. Delegates, which form the basis for .NET events, are much like WinAPI callbacks, but with a few important differences. Randy Rogers and Dave Harms explain.

Posted Wednesday, November 26, 2008

C7 Beta Update: AppGen III

The third C7 AppGen beta release has gone out to third party vendors for compatibility testing. A number of issues have been fixed, including a major memory leak. Dave Harms evaluates this latest release.

Posted Monday, November 24, 2008

Dual or Quad Core For Clarion?

A new release of Clarion is a good excuse as any for a new development box, even if code generation and compiling are already faster. But what processor to choose: dual core or quad core?

Posted Monday, November 17, 2008

Prompting For Time

We use file dialogs for selecting files, color dialogs for selecting colors, and calendars for selecting dates. But what about selecting times? Paul Blais presents a time picker that not only lets you select a block of time but also graphically displays sunset, sunrise and twilight.

Posted Thursday, November 13, 2008

Source Code Library 2008.10.31 Available

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

Posted Wednesday, November 05, 2008

[\[Last 10 articles\]](#) [\[Last 25 articles\]](#) [\[All content\]](#)

Source Code

The ClarionMag Source Code Library

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

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

[More info](#) • [Subscribe now](#)

o » [Understanding Clarion# Delegates](#)

[\[More Clarion & .NET\]](#)

[\[More Clarion 101\]](#)

Latest Free Content

o » [Source Code Library 2008.10.31 Available](#)

[\[More free articles\]](#)

Clarion Sites

Clarion Blogs

o » [John Griffiths](#)

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:



o » [Clarion Tips & Techniques Volume 4 - ISBN 978-0-9784034-09](#)

o » [Clarion Tips & Techniques Volume 3 - ISBN: 0-9689553-9-8](#)

o » [Clarion 6 Tips & Techniques Volume 1 - ISBN: 0-9689553-8-X](#)

o » [Clarion 5.x Tips and Techniques, Volume 1 - ISBN: 0-9689553-5-5](#)

o » [Clarion 5.x Tips and Techniques, Volume 2 - ISBN: 0-9689553-6-3](#)

o » [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.

Copyright © 1999-2008 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Clarion Magazine

Clarion News

[Search the news archive](#)

Easing Into C7: Blog Entries

Bob Foreman has a series of blog posts (nine so far) covering various C7 topics.

Posted Thursday, December 04, 2008

Web Site Grid System

Mark Riffey pointed out this site which provides a grid system for rapid prototyping of web site design elements.

Posted Thursday, December 04, 2008

QuickBooks XML Connect For Clarion

American Riviera Software has a video up on their QuickBooks work in progress. The application in the video is compiled as standalone and is 539k plus a support DLL of 38k.

Posted Thursday, December 04, 2008

Russ Eggen Guest Blog

The UK CUG site now allows guest bloggers, and Russ Eggen is first past the post.

Posted Thursday, December 04, 2008

Sage VAT Tip

Richard Rose provides a heads up for Clarion developers using Sage via CSV import or SDK. Contrary to the official Sage advise of just changing the rate in Sage T1 from 17.5% to 15%, you should create a new tax code say T11 for 15%. Officially T2 is vat for insurance, T3 is for exports and so on. T11 isnt used.

Posted Thursday, December 04, 2008

DMC 1.5.0.6

DMC 1.5.0.6 is now available. Changes include: Use the Viewer on any table then use the query builder to retrieve a subset of the records and to save this subset to the original file (all other records will be deleted); Fixed a small bug in the ASCII reading (improper display of the formats) and also another one when the wrong type of file was selected. The web site has been completely rewritten.

Posted Thursday, December 04, 2008

ClarionFolks C7 Series Part 4

This time Stu adds System Tray functionality via a third party template.

Posted Thursday, December 04, 2008

Lodestar Software Adopts Subscription Model

Beginning 01-Jan-09 Lodestar Software will be changing to an annual subscription plan. Enrollment discounts are available through the end of the year. If you recently purchased one or more Lodestar products you will receive an email shortly. The subscription will provide a license to all Lodestar products and eventually access to a private news server for sharing information and for product support.

Posted Thursday, November 27, 2008

Easing into Clarion 7 - Part Two - Dictionaries

Bob Foreman's blog post and video covers C7 dictionaries.

Posted Wednesday, November 26, 2008

Easing into C7 - Part One - Creating Something New

Bob Foreman has a blog post and video on creating new Clarion 7 solutions.

Posted Wednesday, November 26, 2008

C7 Video Part 3

The third video is up and covers the browse/form paradigm.

Posted Wednesday, November 26, 2008

C7 Video Part 2

In this video Stu Andrews takes a quick look at adding a message button to the app.

Posted Wednesday, November 26, 2008

UK User Group Web Site Revamped

The UK CUG web site has been revamped with DotNetNuke. Feel free to register, send me comments etc.

Posted Wednesday, November 26, 2008

FullRecord 2.20

FullRecord 2.20 is now available. There are numerous new features and fixes.

Posted Wednesday, November 26, 2008

CoolFrames 1.19

CoolFrames 1.19 beta is now available. Please note that the install file requires an unlocking code which is supplied when you purchase CoolFrames.

Posted Wednesday, November 26, 2008

Clarion 7 Video Part 1

Stu Andrews has posted the first of four videos taking a simple look at Clarion 7.

Posted Wednesday, November 26, 2008

Clarion SQL Kickstart

Johan van Zyl has posted a Clarion SQL kickstart, concentrating (for now at least) on MS SQL.

Posted Wednesday, November 26, 2008

Clarion Folklore Podcast #3

The third Clarion Folklore podcast is now available. Topics include conversions, programming entry levels, and other useful stuff.

Posted Tuesday, November 18, 2008

Product Scope 7 Video Tutorial

A video tutorial for Product Scope 7 is now available. Product Scope is used display, create, organize, and research Profile Exchanges including the Clarion Third Party Profile Exchange.

Posted Tuesday, November 18, 2008

Noyantis CommandBars 1.13

Version 1.13 Beta of the Noyantis CommandBars wrapper template has been released. This small release fixes a couple of oversights in the recent v1.12 release. The new version can be downloaded from the Members area using the original download and registration details contained in your sales emails.

Posted Tuesday, November 18, 2008

Noyantis TaskPanel 1.04

Version 1.04 of the Noyantis TaskPanel wrapper template has been released. Modifications include: User definable actions added to item definitions; User definable 'add condition' added to item definitions; Internal update for CommandBars compatibility. Demo available.

Posted Monday, November 17, 2008

Duplicate Symbols Blog Entry

Arnor Baldvinsson has posted a blog entry on the duplicate symbol error showing up in the pre-release of C6 9059.

Posted Monday, November 17, 2008

Noyantis CommandBars 1.12 Beta

Version 1.12 Beta of the Noyantis CommandBars wrapper template has been released. Changes include: New Tabbed Toolbar; User definable actions added to control definitions.; User definable 'add condition' added to control definitions; Clarion menu structure (or part of) can now be specified as bar content along with a visual strategy to use for the content; New method (SetBarIconSize) added to set default icon size for a bar; RibbonBar Group Options Button facility added; Bar Docking settings added along with new method - SetBarDocking; Attach To Parent Control settings added for CommandBarCtrl window control template; Quick Access options can now be added without a System Menu being present; Quick Access options can now be merged with the Tabs when a System menu is not present; New methods added - SetTextValue, SetRadioValue, SetCheckboxValue, GetTextValue, GetRadioValue, GetCheckboxValue; Template Tidyup (both Internal and Interface); Several bug fixes. The new version can be downloaded from the Members area using the original download and registration details contained in your sales emails. Demo updated.

Posted Monday, November 17, 2008

iQ-XML For Clarion 6.3.9059

iQ-XML has been compiled for Clarion 6.3.9059 and is now available. This release works for 9053 through 9059. iQ-XML is a free tool for Clarion developers to add XML functionality to their applications with very little knowledge. It offers

many features not found in Clarion's own XML functions. iQ-XML comes with both Parser and Writer functions. Generate an XML document from a Clarion Queue, Group structure, or using the API's. A novice user can read a complex XML document and fill a Clarion Queue easily. Navigate easily through the XML document, finding nodes and parsing only what you need.

Posted Monday, November 17, 2008

IceTips Utilities 1.1.2324

Build 1.1.2324 of the IceTips Utilities is now available for download. This build includes a few fixes, but the main update is the documentation of the ITUtilities class and the ITString class. The PDF manual is now 227 pages. You must have a valid Gold subscription to be able to download the new release. This release of the IceTips Utilities includes both Clarion 6 and Clarion 7 applications. All IceTips utilities templates seem to register without any issues at all in Clarion 7. The small demo apps that are included with the IceTips Utilities also converted without any problems.

Posted Monday, November 17, 2008

SetupBuilder 6.9 Build 2415

SetupBuilder 6.9 Build 2380 is now available free of charge, to all SetupBuilder customers who have an active SetupBuilder maintenance subscription plan.

Posted Monday, November 17, 2008

Noyantis ShortcutBar 1.11

Version 1.11 Beta of the Noyantis ShortcutBar wrapper template has been released. This release adds an OCX/COM control content type.

Posted Monday, November 17, 2008

Noyantis CalendarPro 1.11

Version 1.11 Beta of the Noyantis CalendarPro wrapper template has been released. The IDispatch Object was not released correctly on exit causing app to GPF on exit under certain circumstances. This has been fixed. The new version can be downloaded from the Members area using the original download and registration details. New demo available.

Posted Monday, November 17, 2008

WindowToWinForm Converter

Randy Rogers has posted a new version of the WindowToWinForm conversion utility. Please see revisions.txt for the changes.

Posted Tuesday, November 11, 2008

Inback 1.8 Codes Sent

New registration codes have been sent to all current users of Inback - check your inbox.

Posted Tuesday, November 11, 2008

Clarion Magazine

Code Snippets In The New IDE

by Dave Harms

Published 2008-11-28

The new Clarion IDE contains a nifty feature called *code snippets*. This is a carryover from the SharpDevelop code base, and in SharpDevelop this feature is called *code templates*. But we all know what real templates are like, and these aren't them.

Still, code snippets are quite useful. In their simplest form they're just a block of text you paste into your code, but you can also include a block of selected text in the generated code, and that block can be repeated as often as you like.

Figure 1 shows the Code Snippets window found under Tools | Options | Coding | Code Snippets- the Clarion snippets are selected.

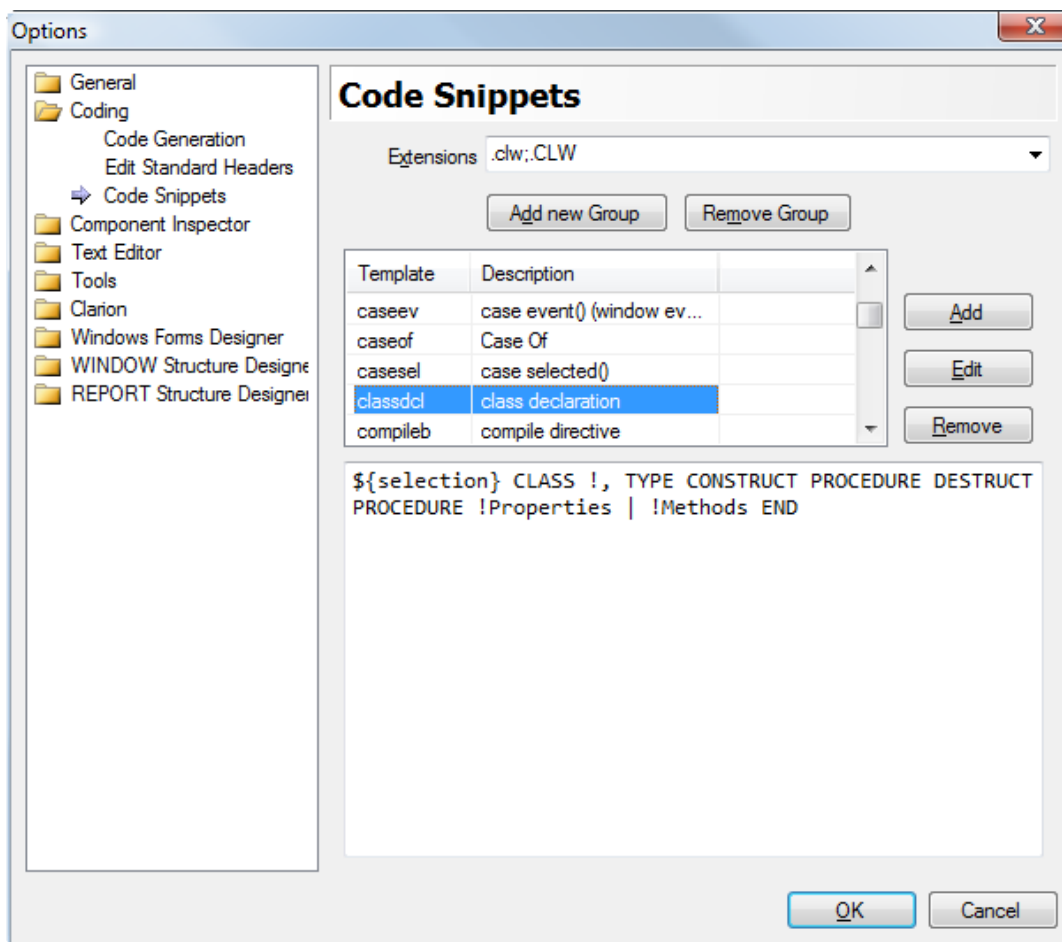


Figure 1. Defining code snippets

The Extensions drop-down list lets you specify different code snippets for different languages. The latest build of Clarion

7 ships with snippets for Clarion, Clarion#, C Sharp, VisualBasic.NET, and HTML. You can add new groups to support additional extensions, and you can add new snippets to any extension or modify the shipping snippets. Snippets are stored in the SharpDevelop-templates.xml file, which on Vista is found in the \Users\<>your user name>\AppData\Roaming\SoftVelocity\Clarion\7.0 directory. That location suggests that the file isn't overwritten when a new version is installed, but if you've made changes you might want to back it up just to be sure.

There are some oddities in the snippets that ship with the current beta release. As you can see in Figure 1, many of the snippets use pipe (|) characters, presumably to indicate line breaks. Since you can edit the templates in a text field and use actual line breaks those characters no longer seem necessary.

Fortunately it's very easy to fix up these snippets. Just select the snippet from the list and type the corrected text. Here's the corrected version of the classdcl snippet from Figure 1:

```

${selection} CLASS !, TYPE
CONSTRUCT PROCEDURE
DESTRUCT PROCEDURE
!Properties
!Methods
END

```

The \${selection} symbol is, as far as I know, the only symbol available to code snippets, and it simply refers to any code that's selected at the time you run the snippet.

To see a snippet in action open a Clarion source file and type

```
MyClass
```

Then select MyClass and press Ctrl-J. The code snippet list appears, as shown in Figure 2.

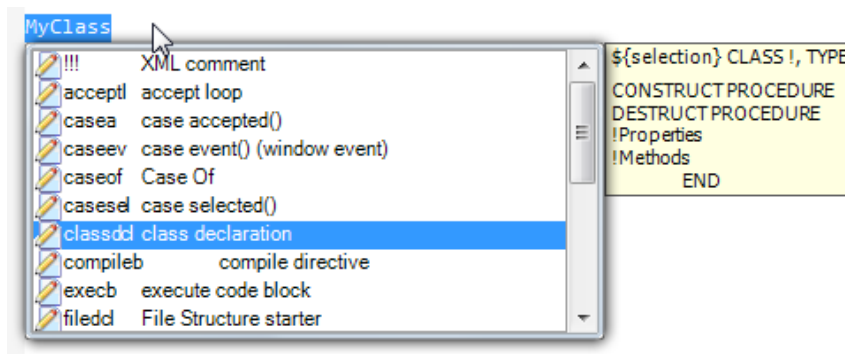


Figure 2. Selecting a code snippet

Select the code snippet and press Enter, or double-click on the snippet. You'll see the result shown in Figure 3.

```

MyClass          CLASS !, TYPE
CONSTRUCT        PROCEDURE
DESTRUCT         PROCEDURE
!Properties
!Methods
                END

```


Figure 3. Code snippet output

Conveniently, the snippet just placed is also automatically formatted.

If you want to get a little fancier you can change the snippet so it generates the class methods as well:

```

${selection} CLASS !, TYPE
CONSTRUCT PROCEDURE
DESTRUCT PROCEDURE
!Properties
!Methods
    END

${selection}.CONSTRUCT PROCEDURE
    CODE

${selection}.DESTRUCT PROCEDURE
    CODE

```

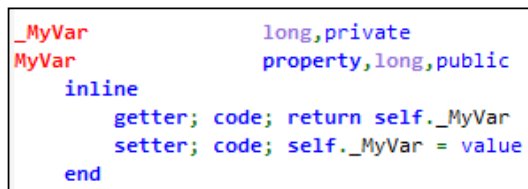
I don't find I use snippets for code I'm already familiar with, such as declaring classes. But I have created a few snippets for Clarion# to handle the generation of class properties. Really, this is something the IDE should (and I hope one day will) do, but for now snippets make the job easier. Here's the snippet code I use to create a Clarion# class property with a LONG data type:

```

_${Selection}      long,private
${Selection}      property,long,public
    inline
        getter; code; return self._${Selection}
        setter; code; self._${Selection} = value
    end

```

This snippet code generates a slightly more compact form of the usual getter/setter code, thanks to the use of continuation semicolons instead of line breaks. Figure 4 shows the generated code.



```

_MyVar      long,private
MyVar      property,long,public
    inline
        getter; code; return self._MyVar
        setter; code; self._MyVar = value
    end

```

Figure 4. A generated LONG property

That's definitely a lot less work than typing out the getter/setter by hand.

Code snippets may be pretty crude compared to Clarion's template technology, but they're handy and very easy to create and modify.

[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 Wednesday, December 03, 2008 by Skip Williams

Code Snippets...yep..kind of just like what Carl Barnes had in CWAssist in 1998? <g> He was just ahead of his time.

[Add a comment](#)

Clarion Magazine

Understanding Clarion# Events

by Dave Harms and Randy Rogers

Published 2008-11-26

Part 1 of this two-part series looked at .NET delegates, which are very similar to the callback procedures familiar to many Clarion developers. But as we explained in that article, delegates aren't always the safest option for component-based development. Delegates assigned to a component can be invoked by code outside the component, which isn't always desirable, and in a multicast situation it's possible to overwrite a delegate assignment done elsewhere.

The solution to both these problems is to modify the delegate to restrict its use to specific circumstances. A delegate so modified is called an *event*.

Defining "event"

There is often confusion over the terms used to describe event-based systems. In one sense an *event* is the something happening in the program that requires a response. The event could be a human-initiated action such as a mouse click or it could be triggered by program logic (for instance, when a process completes).

In the .NET programming sense, an event is a *special kind of delegate* designed to make event handling as safe as possible.

The ClockWithEvent example in the downloadable source is the same as the ClockWithDelegate example, except for a few changes in the ClockWorks.cln source file. Here's the new source:

```
MAP
    TimeChangedDelegate(TimeArgs args),DELEGATE,PUBLIC
END
```

```
using System
```

```
ClockWorks    CLASS,PUBLIC
Run           PROCEDURE
!TimeChanged  TimeChangedDelegate
TimeChanged   EVENT,TimeChangedDelegate
!New method
OnShowEvent  PROCEDURE

end
```

```

ClockWorks.Run    PROCEDURE
  code
  loop 5 TIMES
    System.Threading.Thread.Sleep(1000)
    self.OnShowEvent()
  end

```

```

ClockWorks.OnShowEvent  PROCEDURE
dt      System.DateTime
args    TimeArgs
  code
  if self.TimeChanged <> NULL
    dt = System.DateTime.Now
    args = NEW TimeArgs(dt.Hour,dt.Minute,dt.Second)
    self.TimeChanged(args)
  end

```

The MAP statement is unchanged; to use events you still need the DELEGATE declaration.

The delegate declaration inside the class *has* changed. Before it was

```

TimeChanged      TimeChangedDelegate

```

and now it's

```

TimeChanged      EVENT,TimeChangedDelegate

```

Note the EVENT statement, followed by the delegate type. Remember, TimeChanged is still a delegate, but now it has magical event powers!

Leave the rest of the ClockWorks code aside for a moment and have another look at that misbehaving program code. If you attempt to call the event/delegate directly:

```

works.TimeChanged(t)

```

you'll get this error:

The event '... TimeChanged' can only appear on the left hand side of += or -= (except when used from within the type '... ClockWorks')

That magic event power has now rendered TimeChanged impervious to any attempt to call it directly *except from within the ClockWorks class*. Not only that, but it's no longer possible to do an = assignment. This code:

```

works.TimeChanged = ConsoleView.DisplayTime

```

results in exactly the same error as the previous line of code.

Changing the TimeChanged delegate to the TimeChanged event greatly improves the safety of this feature. Now calling code can only add/remove its own methods, not muck about with anyone else's. And calling code can't interfere with the internal workings of the class presenting the event.

Going back to the ClockWorks class, you'll notice one other significant change. Although not strictly necessary, the code that calls the event/delegate has been moved into its own method called OnShowEvent. It's standard practice to name a method used to trigger an event as "On" plus the event name. Putting the event triggering code in its own method also simplifies calling the event if you need to do so from more than one point in the class code. And, if you use virtual methods, it also opens up the possibility of replacing the event handling code itself in a derived class.

Event parameters

The ClockWithEvent example uses a single parameter for the sake of simplicity, but that actually isn't best practice. By convention, event delegates in the .NET Framework have two parameters, the source that raised the event and the data for the event.

```
MAP
    SecondChangeEventHandler(object sender, SecondChangeEventArgs e)|
    ,DELEGATE,PUBLIC
END
```

If the event handler requires domain specific data then create a class derived from the System.EventArgs class that contains the following:

- A constructor that initializes the domain specific data
- Properties that provide read only access to the data

Here's an example of a time class that conforms to the standard:

```
SecondChangeEventArgs CLASS(System.EventArgs),TYPE,PUBLIC,NETCLASS
    _hour                System.Int32,PRIVATE
    hour                 PROPERTY,System.Int32,PUBLIC,GETONLY
                        INLINE
                        GETTER
                        CODE
                        RETURN SELF._hour
                        END
    _minute              System.Int32,PRIVATE
    minute               PROPERTY,System.Int32,PUBLIC,GETONLY
                        INLINE
                        GETTER
                        CODE
                        RETURN SELF._minute
```

```

                END
_second        System.Int32,PRIVATE
second        PROPERTY,System.Int32,PUBLIC,GETONLY
                INLINE
                GETTER
                CODE
                RETURN SELF._second
                END

Construct      PROCEDURE(System.Int32 hour, System.Int32 minute, |
                System.Int32 second),PUBLIC
                END

SecondChangeEventArgs.Construct PROCEDURE(System.Int32 hour, |
                System.Int32 minute, System.Int32 second)

                CODE
                SELF._hour = hour
                SELF._minute = minute
                SELF._second = second
                RETURN

```

Summary

Delegates are much like callbacks in that they let you plug your own code into someone else's code, but they're also much more flexible and powerful than WinAPI callbacks. With multicasting you can easily attach multiple event handlers, and by converting a delegate to an event you can greatly increase its safety and reliability.

Keep in mind the following differences between events and delegates:

- Delegates are declared in the MAP
- Events are declared in a class
- Every event needs a delegate; delegates can exist without events
- Anyone with access to a delegate can modify it
- Events can only be called directly within the declaring class (or a derived class)
- You can only use the += and -= operators with events
- You can use the =, += and -= operators with delegates

Events and delegates are key to understanding how many .NET applications work, particularly those that interact with the users. But events and delegates are also often employed in situations where no user interface is involved.

Software development is continuing to evolve from monolithic systems to component-based systems that allow for easy interoperability between components. Delegates and events are important tools for building complex yet loosely-coupled systems.

[Download the source](#) (contents of this file are the same as for the [delegates article](#))

[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](#)).

[Randy Rogers](#) is a data processing professional with over 35 years of experience in a wide variety of industries including accounting, municipal government, insurance, printing, and pharmacoconomics. He has a degree in Mathematics from Florida State University and is the president of [Keystone Computer Resources](#). Randy is the author of [ClassViewer](#), a utility for browsing the Clarion class hierarchies. He is also the creator of NetTools, Queue Edit-in-Place, and Screen Capture Tools for Clarion application developers.

Reader Comments

[Add a comment](#)

Clarion Magazine

Edit-In-Place DropLists

by Geoff Thomson

Published 2008-11-28

Using the DropList class with ABC's EditInPlace (EIP) is reasonably trivial - so long as you've done it before. The problem is there's always a first time for doing it. In this article I'll walk through the creation of a simple EIP drop list.

Here's the requirement: you want the user be able to select one of three fixed items, as shown in Figure 1.

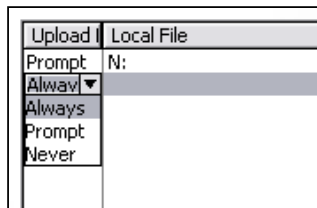


Figure 1. A three item EIP drop list

All of the following steps are specific to the browse, so all the embeds are in the local procedure containing the browse.

Step 1. Declare a queue in the Local Data | Generated Declarations embed point. This queue will contain the drop list options.

```
DirectiveQueue    queue,pre(DQ)
Directive         string(20)
end
```

Step 2. Populate the choice data into the queue. You can use the ThisWindow.init method (somewhere near the end should do, say the Local Objects | ABC Objects | Window Manager | Init embed, about priority 8000)

```
DQ:Directive = 'Always'
add(DirectiveQueue)
DQ:Directive = 'Prompt'
add(DirectiveQueue)
DQ:Directive = 'Never'
add(DirectiveQueue)
```

Step 3. In the Window Formatter (for that procedure) you need to add the update control template (if you haven't done this already). Place the three buttons and hide each one. You won't need them.

Right-click on one of the controls and go to Actions. Tick the 'Use Edit in Place' checkbox. Click the Configure Edit in Place button as shown in Figure 2.

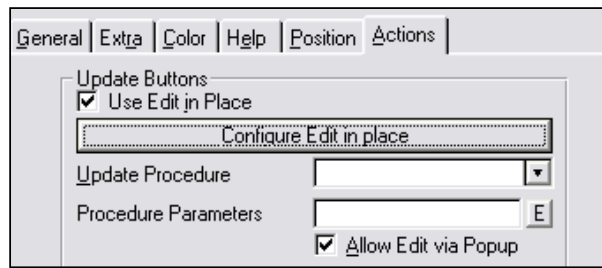


Figure 2. The Configure Edit in place button

Goto the Column Specific tab and click the Insert button.

Enter the field you want to override and check the 'Allow Edit-In-Place' checkbox. Next, uncheck the Use Default ABC:EditEntryClass option and check the Use Application Builder Class option. Now select EditDropListClass from the Base Class drop list (Figure 3).

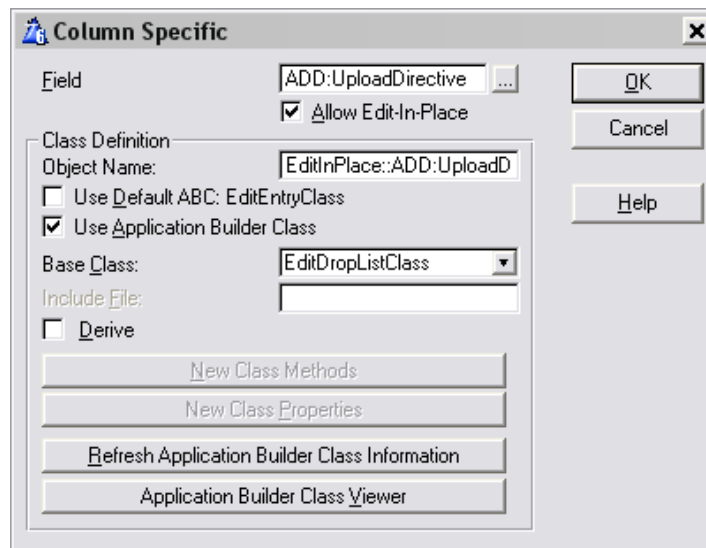


Figure 3. Setting column-specific EIP options

That's it for the Actions on the EditButtons, so press OK a few times to get back to the window formatter.

You'll need to take note of the queue used in the browse (in this case Queue:Browse); you can get this information from the List Properties of the Browse (the From prompt in Figure 4).

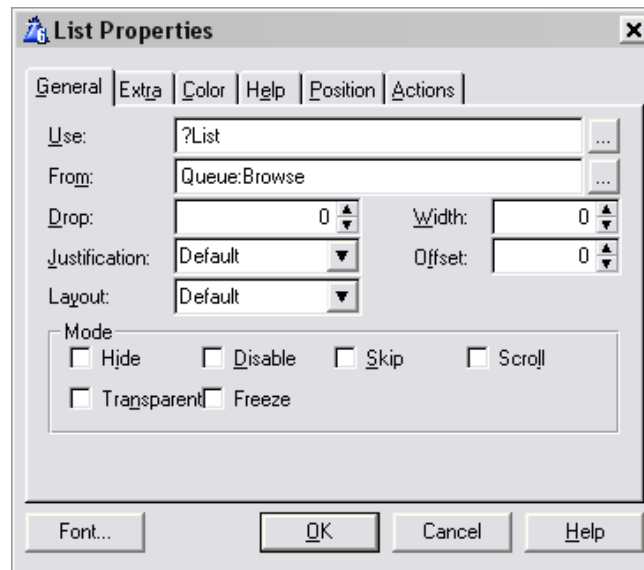


Figure 4. Viewing the From attribute

Save the window changes and get back to the Procedure Properties window.

Step 4. Force the DropDownListClass to use your queue, and get the value that's already saved (if it's there). In the Local Objects | Abc Objects | EIP Field Manager for Browse Using <ListControl> for field <FieldName> (EditDropDownListClass) | Init embed, after the parent call, put the following code:

```
self.FEQ{prop:from} = DirectiveQueue      !This is the queue created in Step 1.
DQ:Directive = Queue:Browse.ADD:UploadDirective !This is the Queue used in the Browse/Listbox.
get(DirectiveQueue,DQ:Directive)
if ~errorcode()
  select(self.FEQ.pointer(DirectiveQueue))  !Determined by the order of the queue.
end
```

Step 5. Save the value selected back into the Browse when the user selects a value. In the Local Objects | ABC Objects | EIP Field Manager for Browse Using <ListControl> for field <FieldName> (EditDropDownListClass) | TakeAccepted embed (after the parent call), put the following:

```
get(DirectiveQueue,choice(self.feq))
Queue:Browse.ADD:UploadDirective = DQ:Directive
put(Queue:Browse)
```

And that's it. The browse class handles the updating of the file.

Geoff Thomson started out in the electronics field, first as a technician, then as a test engineer before taking on a management role where he began dabbling in Clarion for Windows (version 2). Six months later he found himself with two job offers: one for TopSpeed South Africa as a senior support programmer, and one for Capesoft. After much deliberation he joined Capesoft, primarily in the hardware department (air-conditioning controls and time clocks). It was not long after that that he began to get more and more involved in the accessories side of the business. Geoff is now a partner in

CapeSoft, and his responsibilities include FM3, Replicate, HyperActive, MessageBox, SendTo, HotDates and WinEvent, to name a few. Out of the office, Geoff spends time with his five children and lovely wife Natalie, whom he married in 1995. He enjoys sport (particularly cricket), reading good books, games (bridge, canasta, cluedo, monopoly, etc) and walking in the mountains, a number of which are within a few minutes from his home in Hout Bay, Cape Town. Because of his love for Jesus, he is very involved in his church, particularly on the music side. Geoff plays a number of musical instruments such as the piano, keyboards, guitar, cello, harmonica, etc. Geoff and Natalie share a love for living education (based on the Charlotte Mason education philosophy). Besides home-educating their own children, they have been involved in training home-educators and teachers in the greater Cape region as well as in other countries such as Malawi.

Reader Comments

[Add a comment](#)

Clarion Magazine

#FINDing Values In Prefixed Structures

by Dave Harms and Lee White

Published 2008-11-28

In the latest build of C7, SoftVelocity introduced a number of new template symbols including %LocalDataID, %LocalDataFull, %ModuleDataID, %ModuleDataFull, %GlobalDataID and %GlobalDataFull.

It's all Lee White's fault.

A little while ago Lee discovered a problem with using a prompt type of FIELD with local, module, or global data. The string that gets put into the symbol associated with the prompt retains the prefix. That's fine unless you want to go back and look up the selected item (in local, module or global data) to get additional information. To do that you need to use the #FIX statement. Actually that works fine if you're #FIXing a field in a table since tables have to have a prefix. But using #FIX to look up other data that's part of a prefixed QUEUE or GROUP doesn't work, because #FIX doesn't see the prefix associated with the QUEUE or GROUP.

This problem affects all releases of Clarion through C6 9058. Reportedly the problem has been fixed in C6 9059, but there's no independent confirmation of that yet, or whether the fix also involves the new template symbols.

Consider this local data:

```
DataField    STRING(10)
XGroup       GROUP,PRE(XGrp)
DataField    STRING(20)
            END
```

You can't #FIX(%LocalData, 'XGrp:DataField'); #FIX doesn't know how to resolve the string 'XGrp:DataField' to the prefixed GROUP.

You could strip the prefix, which was Lee's first thought and attempt, but then you get the properties for the STRING (10) iteration, not the one in the group.

In C6, the following workaround will get you by until C7 arrives with its additional symbols.

The template shown below sets up a hidden MULTI prompt to mimic access to the GROUP or QUEUE. Hidden prompts are useful for all kinds of template programming tasks, mainly because they provide a way of declaring persistent data. So don't think of this MULTI as a prompt, think of it as a data structure.

The %BuildFieldList group stores all local, module and global data in the symbols declared in the MULTI. That way you only need one #FIND(%LWFieldName,'PRE:field') or #FIND(%LWFieldName,%MySymbol) statement to search all the data.

Here's the template code:

```
#TAB('Hidden'),WHERE(%FALSE)
#BUTTON('Field Queue'),MULTI(%LWFieldQueue,%LWFieldName)
```

```
#PROMPT(",@s255), %LWFieldName
#PROMPT(",@s255), %LWFieldStatement
#PROMPT(",@s255), %LWFieldPicture
#ENDBUTTON
#ENDTAB

#GROUP(%BuildFieldList)
#DECLARE(%EndCounter, LONG)
#DECLARE(%StructPrefix)
#FREE(%LWFieldQueue)
#SET(%EndCounter, 0)
#SET(%StructPrefix, ")
#!
#! add global data to field list
#!
#FOR(%GlobalData)
  #IF( INSTRING('PRE(', UPPER(%GlobalDataStatement), 1, 1) )
    #IF( %EndCounter = 0 )
      #SET(%StructPrefix, EXTRACT(%GlobalDataStatement, 'PRE', 0))
      #IF(%StructPrefix <> ")
        #SET(%StructPrefix, %StructPrefix & ':')
      #ENDIF
    #ENDIF
    #SET(%EndCounter, %EndCounter+1)
  #CYCLE
#ENDIF
#IF( UPPER(%GlobalDataStatement) = 'END' )
  #SET(%EndCounter, %EndCounter-1)
  #IF(%EndCounter <= 0)
    #SET(%EndCounter, 0)
    #SET(%StructPrefix, ")
  #ENDIF
  #CYCLE
#ENDIF
#ADD(%LWFieldQueue, ITEMS(%LWFieldQueue)+1)
#SET(%LWFieldName, %StructPrefix & %GlobalData)
#SET(%LWFieldStatement, %GlobalDataStatement)
#SET(%LWFieldPicture, ")
#ENDFOR
#SET(%EndCounter, 0)
#SET(%StructPrefix, ")
```

```
#!  
#! add module data to field list  
#!  
#FOR(%ModuleData)  
  #IF( INSTRING('PRE(', UPPER(%ModuleDataStatement), 1, 1) )  
    #IF( %EndCounter = 0 )  
      #SET(%StructPrefix, EXTRACT(%ModuleDataStatement,'PRE',0))  
      #IF(%StructPrefix <> "  
        #SET(%StructPrefix, %StructPrefix & ':')  
      #ENDIF  
    #ENDIF  
  #SET(%EndCounter,%EndCounter+1)  
  #CYCLE  
#ENDIF  
#IF( UPPER(%ModuleDataStatement) = 'END' )  
  #SET(%EndCounter,%EndCounter-1)  
  #IF(%EndCounter <= 0)  
    #SET(%EndCounter,0)  
    #SET(%StructPrefix, "  
  #ENDIF  
  #CYCLE  
#ENDIF  
#ADD(%LWFieldQueue,ITEMS(%LWFieldQueue)+1)  
#SET(%LWFieldName, %StructPrefix & %ModuleData)  
#SET(%LWFieldStatement, %ModuleDataStatement)  
#SET(%LWFieldPicture, "  
#ENDFOR  
#SET(%EndCounter,0)  
#SET(%StructPrefix, "  
#!  
#! add local data to field list  
#!  
#FOR(%LocalData)  
  #IF( INSTRING('PRE(', UPPER(%LocalDataStatement), 1, 1) )  
    #IF( %EndCounter = 0 )  
      #SET(%StructPrefix, EXTRACT(%LocalDataStatement,'PRE',0))  
      #IF(%StructPrefix <> "  
        #SET(%StructPrefix, %StructPrefix & ':')  
      #ENDIF  
    #ENDIF  
  #SET(%EndCounter,%EndCounter+1)  
  #CYCLE
```

```

#ENDIF
#IF( UPPER(%LocalDataStatement) = 'END' )
  #SET(%EndCounter,%EndCounter-1)
  #IF(%EndCounter <= 0)
    #SET(%EndCounter,0)
    #SET(%StructPrefix, "")
  #ENDIF
#CYCLE
#ENDIF
#ADD(%LWFieldQueue,ITEMS(%LWFieldQueue)+1)
#SET(%LWFieldName, %StructPrefix & %LocalData)
#SET(%LWFieldStatement, %LocalDataStatement)
#SET(%LWFieldPicture, %LocalDataPicture)
#ENDFOR
#RETURN

```

Implement the code as needed with #ATSTART and #PREPARE sections:

```

#ATSTART
  #INSERT(%BuildFieldList)
#ENDAT
#PREPARE
  #INSERT(%BuildFieldList)
#ENDPREPARE

```

Another useful #GROUP

Here's another #GROUP Lee uses as a lookup function that handles file fields and variables; it also handles dimensioned field lookups. This provides a replacement for most uses of #FIND(); it tries to find the passed variable in a file field and if that fails it tries to look it up in the MULTI.

```

#GROUP(%FindField,%Incoming)
#DECLARE(%ReturnValue)
#SET(%ReturnValue,%FALSE)
#IF( %Incoming )
  #SET(%WorkString,CLIP(LEFT(%Incoming)))
  #IF( INSTRING('[',%WorkString,1,1) )
    #!
    #! handle DIMmed fields by removing dimension before lookup
    #!
    #SET(%WorkString, SUB(%WorkString, 1, (INSTRING('[',%WorkString,1,1))-1 ))
  #ENDIF
  #FIND(%Field,%WorkString)
  #IF( %Field )

```

```

    #SET(%ReturnValue,% TRUE)
#ENDIF
#IF( NOT %ReturnValue )
    #FIND(%LWFieldName,% WorkString)
    #IF( %LWFieldName )
        #SET(%ReturnValue,% TRUE)
    #ENDIF
#ENDIF
#ENDIF
#RETURN( %ReturnValue )

```

Here's how you'd typically use this group:

```
#SET(%Flag,%FindField(%FieldToFind))
```

or

```
#IF( %FindField(%FieldToFind) )
    #! do stuff
#ENDIF
```

The C7 solution

In C7 (and possibly in C6 9059) the solution is to use the aforementioned new symbols: %LocalDataFull, %ModuleDataFull and %GlobalDataFull. These symbols return the fully prefixed label for each item. There are also new %GlobalDataID, %ModuleDataID and %LocalDataID symbols. %ModuleDataID and %LocalDataID, at least, are synonyms for %ModuleData and %LocalData. None of the previously existing symbols have changed so no templates should be broken by these changes.

[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](#)).

Lee White's introduction to computer languages began during his sophomore year in high school, through an open class at the University of Alabama, c1969. His first language was Fortran 4 with Watfor. Introduced to Clarion 2.1 in 1990, he has remained a steadfast Clarion user and supporter. He is often better known for his contributions to the etc conferences and [third party products](#).

Reader Comments

[Add a comment](#)

Clarion Magazine

Understanding Clarion# Delegates

by Dave Harms and Randy Rogers

Published 2008-11-26

Events and Delegates are two important .NET concepts that often generate a lot of confusion. In this article we will explain the mechanics of events and delegates, and we will illustrate how events and delegates make it possible to create loosely-coupled applications with more easily reusable components.

Events, delegates and callbacks

Events and delegates are often described as two different things, but really they're just variations on the old idea of a *callback procedure*. A callback is simply a chunk of your own code that you pass to somebody else's code, and then that code "calls back" into your code. Callbacks are commonly used to provide visual feedback to a user during a long process, such as an FTP file download. If you've ever done any Windows event subclassing you've also used callbacks to have your own code process window events that aren't otherwise available via the ACCEPT loop.

And, if you've worked with Windows API callbacks, you probably know how easy it is to crash your app with a bad implementation. Callbacks aren't particularly safe, as code goes.

In .NET, on the other hand, callbacks are quite safe and are an integral part of many applications. We'll begin our discussion with delegates, which are the basic .NET callback implementation, and then move on to the more restrictive case of events.

Delegates - a procedural view

Clarion#, like Clarion, is a hybrid language with both procedural and object-oriented syntax. We'll begin with a procedural example and then move on to object-oriented examples.

NOTE: The downloadable source code includes five sample projects. Rather than loading each project's solution individually you may find it easier to load the ClarionMag.Articles.EventsAndDelegates solution which contains all five projects.

Here's the code for the ProceduralDelegate project:

```
PROGRAM
```

```
    NAMESPACE(ClarionMag.Articles.EventsAndDelegates.ProceduralDelegate)
```

```
    USING System
```

```
    MAP
```

```
        SimpleDelegate(),DELEGATE,PUBLIC
```

```
        myFunc()
```

```
END
```

```
MyDelegateReference    SimpleDelegate
```

```
CODE
```

```
MyDelegateReference = myFunc
```

```
IF MyDelegateReference <> NULL
```

```
    MyDelegateReference()
```

```
END
```

```
Console.WriteLine('Done - press any key')
```

```
Console.ReadKey()
```

```
RETURN
```

```
myFunc    PROCEDURE()
```

```
CODE
```

```
Console.WriteLine('myFunc was called by MyDelegateReference ...')
```

```
RETURN
```

Inside the MAP you'll see this declaration:

```
SimpleDelegate(),DELEGATE,PUBLIC
```

The DELEGATE keyword marks this declaration as a delegate. Here's what the Clarion help has to say:

"A **DELEGATE** is equivalent to function pointers (i.e., callback functions), but object-oriented and type safe. Type safe means that if the function signature implemented by the Client differs in terms of number of parameters or parameter types or return values as opposed to the method signature, it is going to safely raise an error."

SimpleDelegate isn't a procedure, and isn't by itself a reference to a procedure. Rather, it's a data type. And as with other data types you need to create an instance in order to use it:

```
MyDelegateReference    SimpleDelegate
```

Now you have a reference called MyDelegateReference, and it is of type SimpleDelegate. That means that you can assign any method or procedure to MyDelegateReference as long as that method or procedure has the same prototype as SimpleDelegate.

As it turns out, there is a procedure in this program with the same prototype (no parameters, no return type), and it's called MyFunc. MyFunc can be assigned to MyDelegateReference one of two ways:

```
MyDelegateReference = NEW SimpleDelegate(myFunc)
```

or

```
MyDelegateReference = myFunc
```

Both approaches generate exactly the same IL code. And once you've done the assignment, calling MyDelegateReference results in a call to MyFunc. Figure 1 shows the program output:

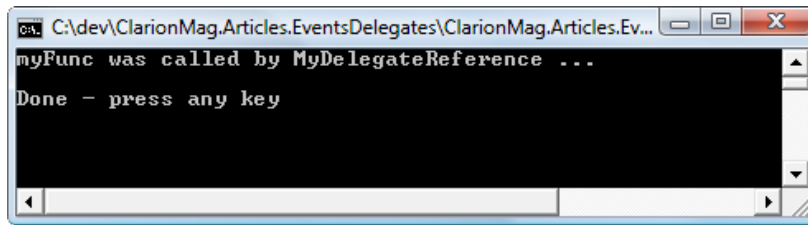


Figure 1. Calling MyFunc as a delegate

As you can see, the call to MyDelegateReference results in MyFunc writing a line of text to the console output.

Although you can still write procedural code in Clarion#, under the hood that code is implemented as object-oriented .NET IL code. So you might as well skip the middleman and go straight to OOP.

An object-oriented delegate

Here's the source code for the SimpleDelegate project. The declaration of SimpleDelegate in the MAP statement is identical to the procedural version, but instead of a MyFunc procedure there are now several class methods that are assigned, in sequence, to the method.

PROGRAM

NAMESPACE(ClarionMag.Articles.EventsAndDelegates.SimpleDelegate)

MAP

SimpleDelegate(),DELEGATE,PUBLIC

END

USING System

```
MyClass      class
DoTask       procedure
Main         procedure
MyDelegateReference SimpleDelegate
MyDelegateMethodA procedure
MyDelegateMethodB procedure
            end
```

mc MyClass

code

mc = new MyClass()

mc.Main()

```

MyClass.DoTask  procedure
  code
  loop 10 TIMES
    self.MyDelegateReference()
  end
  Console.WriteLine('DoTask finished - press any key')
  Console.ReadKey()

MyClass.Main  procedure
  code
  ! Either syntax works - long form used here.
  self.MyDelegateReference = new SimpleDelegate(self.MyDelegateMethodA)
  !self.MyDelegateReference = self.MyDelegateMethodA
  self.DoTask
  Console.WriteLine()
  ! Either syntax works - short form used here.
  !self.MyDelegateReference = new SimpleDelegate(self.MyDelegateMethodB)
  self.MyDelegateReference = self.MyDelegateMethodB
  self.DoTask

MyClass.MyDelegateMethodA  procedure
  code
  Console.WriteLine('MyDelegateMethodA called')

MyClass.MyDelegateMethodB  procedure
  code
  Console.WriteLine('MyDelegateMethodB called')

```

In the previous example the delegate reference was a global variable; in this example the reference is contained within the class.

The execution sequence goes like this:

1. The program startup CODE section creates an instance of MyClass and calls MyClass.Main()
2. The Main method assigns MyDelegateMethodA to the delegate reference and calls the DoTask method.
3. The DoTask method calls MyDelegateReference which is a now reference to MyDelegateMethodA
4. The Main method repeats steps 2 and 3 using MyDelegateMethodB

Figure 2 shows the output from SimpleDelegate.

```

C:\dev\ClarionMag.Articles.EventsDelegat...
MyDe legateMethodA called
MyDe legateMethodA called
MyDe legateMethodA called
MyDe legateMethodA called
MyDe legateMethodA called
MyDe legateMethodA called
MyDe legateMethodA called
MyDe legateMethodA called
MyDe legateMethodA called
MyDe legateMethodA called
DoTask finished - press any key

MyDe legateMethodB called
MyDe legateMethodB called
MyDe legateMethodB called
MyDe legateMethodB called
MyDe legateMethodB called
MyDe legateMethodB called
MyDe legateMethodB called
MyDe legateMethodB called
MyDe legateMethodB called
MyDe legateMethodB called
DoTask finished - press any key

```

Figure 2. Two delegates called in succession

This is a very simple example with everything in one class. But there's really no point to having delegates in a single class like this since you could more easily call the desired methods directly. All this example demonstrates is the mechanism; we'll describe a more realistic example shortly. But first, a word about multi-casting.

Multi-cast delegates

Multi-casting is the ability to assign more than one method to a delegate. We won't reproduce all the code for the MulticastDelegate example here as it's almost identical to SimpleDelegate. The only material change is in the Main method:

```

MyClass.Main    procedure
    code
    self.MyDelegateReference = self.MyDelegateMethodA
    ! Add a second method to the delegate
    self.MyDelegateReference += self.MyDelegateMethodB
    self.DoTask

```

Now there is just a single call to DoTask, but more importantly the second reference assignment has been changed. Instead of

```
self.MyDelegateReference = self.MyDelegateMethodB
```

the code is now

```
self.MyDelegateReference += self.MyDelegateMethodB
```

The = sign on the second assignment has been changed to += (actually they could both be +=). That += says "assign this method to the delegate, but don't get rid of any existing assignments." The result is that the output now shows both methods firing on each call to self.MyDelegateReference (Figure 3).

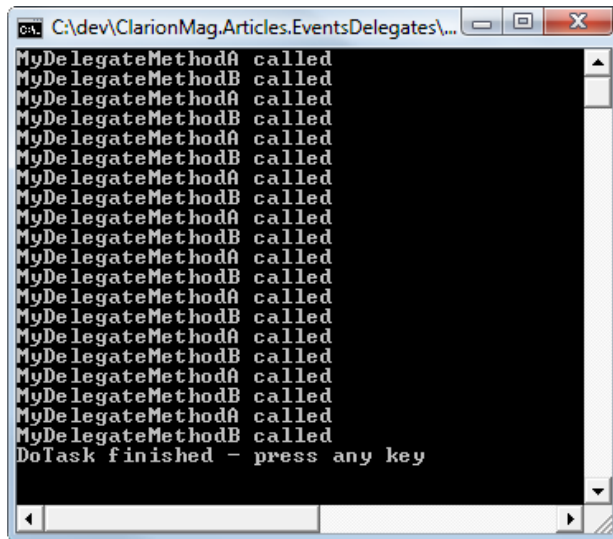


Figure 3. Two delegates called in a multicast configuration

Besides the = operator to assign a method and the += operator to add a method to any existing methods, you can use the -= operator to remove a method.

The only other bit that's been added to MulticastDelegate is the error checking around the use of the reference itself:

```
if self.MyDelegateReference <> NULL
    self.MyDelegateReference()
end
```

Delegates vs virtual methods and interfaces

By now you should be getting the idea that delegates are a nifty way to plug one object into another object. In fact, delegates are reminiscent of virtual methods and interfaces, both of which make it easier to use classes like building blocks that can be assembled in various ways. But there's a very important difference between delegates and virtual methods/interfaces.

Virtual methods require you to implement a derived class, and interfaces require you to implement a class that conforms to the interface specification. Delegates, however, only require you to implement a single method that matches a specified signature.

Both virtual methods and interfaces imply a tighter coupling between your code and the component you want to use. Delegates keep that coupling as loose as possible. And neither virtual methods nor interfaces explicitly support multicasting.

A more realistic scenario

It's time for that "more realistic" scenario we alluded to earlier. But there's an important caveat to the following example. Although it shows how you might use delegates in a real-world situation, it is also intended to illustrate some of the pitfalls of delegates. Conveniently, those pitfalls are fully addressed by *events*, which we'll discuss in the next installment.

If you do any amount of searching the Internet for information on delegates and events, you're almost sure to come across a clock-based example. So here's our Clarion# version.

Clock concepts

The clock example is predicated on the idea that it's better to build systems out of building blocks than out of a

single, monolithic piece of code. We already do that kind of thing by breaking apps up into procedures, into classes etc. But the idea here is to consider how some *thing* in your program might be used in different ways.

A clock is something that keeps time; it also displays time. But if you're talking about a software clock you might want to "display" that time in different ways. And the term "display" doesn't necessarily mean showing the time on the screen. It might mean that; it might also mean informing another system of the time, or writing the time to a log file, or doing any number of activities with time information.

The clock example splits up the clock into functional units. A class called ClockWorks handles the actual timekeeping, and additional classes can be "plugged in" via delegates to provide the desired display of time information.

For demonstration purposes the ClockWithDelegate example application uses two classes to mimic different approaches to showing time. One is called ClockConsoleView and the other is ClockLogFileView. Yes, both of these write information out to the console. We realize that. Suspend your disbelief, and pretend that they're doing very different things.

Here's ClockConsoleView:

```
ClockConsoleView  class
  DisplayTime      procedure(TimeArgs args)
                    end

ClockConsoleView.DisplayTime  procedure(TimeArgs args)
  code
  Console.WriteLine('ClockConsoleView: time is ' & args.hour |
    & ':' & args.minute & ':' & args.second)
```

ClockLogFileView does more or less the same thing, but pretends it is writing to a log file.

You'll notice that the DisplayTime method takes a TimeArgs parameter. TimeArgs is a very simple class with three properties: hour, minute, second. We won't bother listing it here; you can find it in TimeArgs.cln.

There are two remaining source files: ClockWorks.cln and Program.cln. First the program code:

```
works      ClockWorks
consoleView  ClockConsoleView
logFileView  ClockLogFileView
  code
  works = new ClockWorks()
  consoleView = new ClockConsoleView()
  logFileView = new ClockLogFileView()
  works.TimeChanged += consoleView.DisplayTime
  works.TimeChanged += new TimeChangedDelegate(logFileView.DisplayTime)
  works.Run()
```

For the moment don't worry too much about how ClockWorks actually does its thing. The only thing you really need to know is that ClockWorks has a delegate called TimeChanged which matches the signatures of the DisplayTime methods in the view classes.

As you can see, the code first creates a ClockWorks instance, then instances of each of the view classes (the order here doesn't matter). Next, the code assigns the two DisplayTime methods to the TimeChanged delegate using the += operator. And finally the code calls the works.Run() method, which results in the output shown in Figure 4.

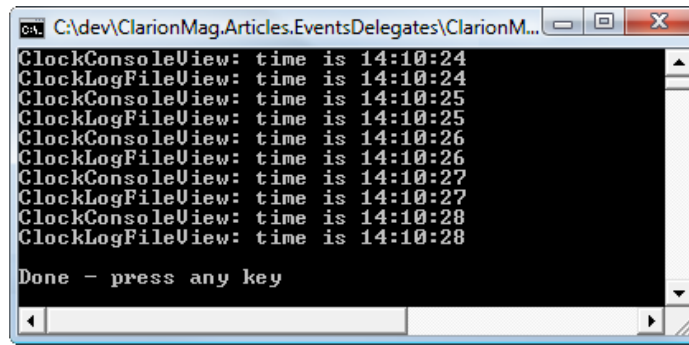


Figure 4. Passing an argument to two delegates

Now, what's inside ClockWorks? Here's the listing:

```

MAP
    TimeChangedDelegate(TimeArgs args),DELEGATE,PUBLIC
END

using System

ClockWorks    CLASS,PUBLIC
Run           PROCEDURE
TimeChanged   TimeChangedDelegate
              end

ClockWorks.Run PROCEDURE
dt            System.DateTime
args         TimeArgs
code
loop 5 TIMES
    System.Threading.Thread.Sleep(1000)
    if self.TimeChanged <> NULL
        dt = System.DateTime.Now
        args = NEW TimeArgs(dt.Hour,dt.Minute,dt.Second)
        self.TimeChanged(args)
    end
end
end

```

First, note the MAP statement containing the delegate declaration, and the corresponding class member called TimeChanged. The prototype for TimeChangedDelegate must match the prototype for any methods assigned to TimeChanged via the = or += operators.

Now have a look at the code for the Run method. The loop executes five times, with a one second wait on each loop. Provided at least one method has been added to TimeChanged, the code creates an instance of TimeArgs, loads it up with the current system time and passes the TimeArgs instance to TimeChanged, which means that TimeArgs gets passed to

any method that's been added to the TimeChanged delegate.

The beauty of delegates over, say, interfaces and virtual methods, is that the code you write to take advantage of a delegate only needs to match up that one method signature. You don't have to work out how to massage your code into somebody else's idea of a class structure.

But there's a dark side to delegates. Assume for a moment that ClockWorks is a component provided to you by someone else. You rely on ClockWorks to provide your code with accurate time information.

Now consider the following version of the program code:

```
works      ClockWorks
consoleView  ClockConsoleView
logFileView  ClockLogFileView
t          TimeArgs
code
works = new ClockWorks()
consoleView = new ClockConsoleView()
logFileView = new ClockLogFileView()
works.TimeChanged += consoleView.DisplayTime
works.TimeChanged += new TimeChangedDelegate(logFileView.DisplayTime)
works.Run()
Console.WriteLine()
Console.WriteLine('Okay, now let's mess things up a little...')
Console.WriteLine()
t = new TimeArgs()
t.hour = 33
t.minute = 88
t.second = 99
works.TimeChanged(t)
Console.WriteLine('Whoops, is that the right time?')
Console.WriteLine()
works.TimeChanged = ConsoleView.DisplayTime
works.TimeChanged(t)
Console.WriteLine('Why did I only get one delegate called now?')
Console.WriteLine()
Console.WriteLine('Done - press any key')
Console.ReadKey
```

This code intentionally causes the kind of trouble you might cause unintentionally. Figure 5 shows the output.

```

C:\dev\ClarionMag.Articles.EventsDelegates\ClarionM...
ClockConsoleView: time is 14:12:14
ClockLogFileView: time is 14:12:14
ClockConsoleView: time is 14:12:15
ClockLogFileView: time is 14:12:15
ClockConsoleView: time is 14:12:16
ClockLogFileView: time is 14:12:16
ClockConsoleView: time is 14:12:17
ClockLogFileView: time is 14:12:17
ClockConsoleView: time is 14:12:18
ClockLogFileView: time is 14:12:18

Okay, now let's mess things up a little...

ClockConsoleView: time is 33:88:99
ClockLogFileView: time is 33:88:99
Whoops, is that the right time?

ClockConsoleView: time is 33:88:99
Why did I only get one delegate called now?

Done - press any key

```

Figure 5. Problems in delegateland

By creating your own `TimeArgs` instance and calling `TimeChanged` directly you've caused the view methods to record invalid data. Clearly there's no such time as 33:88:99, at least not in normal usage. The problem is `TimeChanged` isn't private to `ClockWorks`. In fact, `TimeChanged` can't be private or else you wouldn't be able to assign your methods to the delegate.

The second problem is that you can overwrite `TimeChanged`. The code:

```
works.TimeChanged = ConsoleView.DisplayTime
```

replaces the previous two method assignments with just one method assignment. Now you're only getting a single line of output for each call to `TimeChanged`. This may not be a big problem if your code is the only code that uses `ClockWorks`, but what if `ClockWorks` is part of a larger library and some of that code installs a method in `TimeChanged`? Now you've unknowingly obliterated that call.

Although delegates are incredibly powerful, they're not always that safe to use. Happily, .NET has this nifty little thing called an *event*, which is really just a dressed up version of a delegate. We'll cover those in the [next installment](#).

[Download the source](#) (contents of this file are the same as for the [events article](#))

[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](#)).

[Randy Rogers](#) is a data processing professional with over 35 years of experience in a wide variety of industries including accounting, municipal government, insurance, printing, and pharmacoconomics. He has a degree in Mathematics from Florida State University and is the president of [Keystone Computer Resources](#). Randy is the author of [ClassViewer](#), a utility for browsing the Clarion class hierarchies. He is also the creator of [NetTools](#), [Queue Edit-in-Place](#), and [Screen Capture Tools](#) for Clarion application developers.

Reader Comments

[Add a comment](#)

Clarion Magazine

C7 Beta Update: AppGen III

by Dave Harms

Published 2008-11-24

First off, my apologies for the lateness of this update. I got a little too busy last week and, among other things, got sidetracked by some image display issues. One of the problems I'm encountering with C7 has nothing to do with C7 itself; rather, it's that almost any screen shot of C7 is much bigger than the C6 equivalent, and usually too big for placement in an article without first resizing. But resized images are less readable, so I generally include a link to a full size image. That's always been a hassle, and with more images coming in at the larger sizes I've started looking for a way to conveniently zoom images in-place. If you have any suggestions, particularly for pure CSS solutions, let me know. But I have to warn you I've experimented with a bunch of options and so far I'm unimpressed with the possibilities.

Okay, back to AppGen. SoftVelocity released the third beta build last week, which is in keeping with a build-every-two-weeks schedule. I know a lot of CSP folks are anxious to get their hands on the AppGen, and overall I think SV is making steady progress toward that goal. Here's hoping for a wider release by Christmas (that's not a prediction!).

Whereas previous builds generated a lot of beta newsgroup traffic in short order, things have been a lot quieter following this build and the discussions that have been happening often have more to do with usability issues than with critical bugs. That's not to say there aren't still some major issues, there certainly are. But there have been a number of important fixes as well, and not just to third party template compatibility problems.

Memory leak fixed

The biggest news in this build is the resolution of a rather large memory leak. As I reported [last time](#), the previous two builds had a serious leak to the point where it was problematic, sometimes impossible, to load large multi-DLL solutions. That leak has been plugged and memory usage is now much more reasonable. Previously, each time I loaded and generated the DLLTutor solution I'd see several hundred megs of RAM eaten up until at somewhere around a gig of RAM usage the IDE would crash. Now the IDE stays at around 175-200 megs the whole time.

That may still seem like a lot of memory to some folks. And C7 is definitely a little fatter than Visual Studio; at startup it weighs in (on my machine) at 28 megs of RAM which is a little more than twice what VS takes at startup. Of course, C7 probably has a lot of debug code loaded so it may not be a fair comparison. Load up an app and the code generation overhead kicks in; a 50 procedure app bumps memory usage up to 130 megs. Despite the massive memory usage, C7 is, as previously noted, far faster at generating and compiling than C6.

(On my machine, at least, C7 is one of the bigger programs in terms of memory usage, but by no means the biggest. That honor usually goes to Thunderbird, my email, news and RSS client. TB starts out at 200 megs of RAM and can easily grow to double that if I leave it running for a day. I've seen Firefox approaching that kind of memory usage, although in both cases the problem may be down more to add-ins than the core product itself.)

Interestingly, there's been some discussion about how that memory leak could occur given that the new IDE is a .NET application. Isn't .NET supposed to automatically clean up unused memory? Yes and no.

What we think of as "normal" .NET code is "managed" - that is, you can create objects as you need them, and the

garbage collection system will detect when those objects are no longer needed and will clean them up. But you can still induce a leak, if that is the right term, by hanging onto the object reference when you no longer need it. Is it a leak? I guess it depends on your definition. There's been no confirmation from SV, but this is a plausible explanation for the leak in previous builds.

As well, most .NET applications also call some "unmanaged" native code. And if that code leaks... well, you get the idea.

There's also a potential for problems with some system resources. Let's say you're doing some drawing (perhaps for a custom control) using the System.Drawing classes. Among other things you'll need to get brushes and pens, which are system resources with a finite supply. Here's a snippet from [Enhancing The RoundedPanel Control](#):

```
fillBrush = new SolidBrush(self.forecolor)
e.Graphics.FillPath(fillbrush,p)
fillBrush.Dispose()
framePen = NEW Pen(SELF.BorderColor, SELF.BorderWidth)
e.Graphics.DrawPath(framePen, p)
framePen.Dispose()
```

Note that after creating the brush and the pen I manually Dispose of both of those resources. Yes, the garbage collection code will eventually get around to doing this if you forget, but if you're in a situation where you're using a lot of these system resources you could run out before the garbage collection system frees up the ones you're no longer using.

Major bugs not yet fixed

Not all the template issues uncovered in testing have been fixed. For instance, one vendor's templates, previously unregistrable, can now be registered but crash the IDE whenever a field lookup button is pressed. SV is working on that problem. There are a number of other template-related issues although the core templates seem to working fairly well in most circumstances.

You can run into troubles when modifying applications: for instance, canceling a procedure extension addition doesn't really cancel it.

Another issue is embeditor handling. As I've [previously reported](#), when you encounter a compile error in embedded code and you double-click on the error in the output window you're taken to the generated source, not the embed. I was hoping it would be fixed for this release, but it's not. As I noted in the previous report this isn't as painful a problem as it is in C6 since you can have the source window visible alongside the embeditor if you wish. But it's something that does need to be fixed.

I don't have access to all the bug reports as a number of these are marked as private by submitters, so no doubt there are some other critical issues that I'm not aware of.

New stuff

Some terminology has been cleaned up for this release. As I noted in [Understanding The C7 Build System](#), the project pad has been renamed to Solution Explorer, as shown in Figure 1.

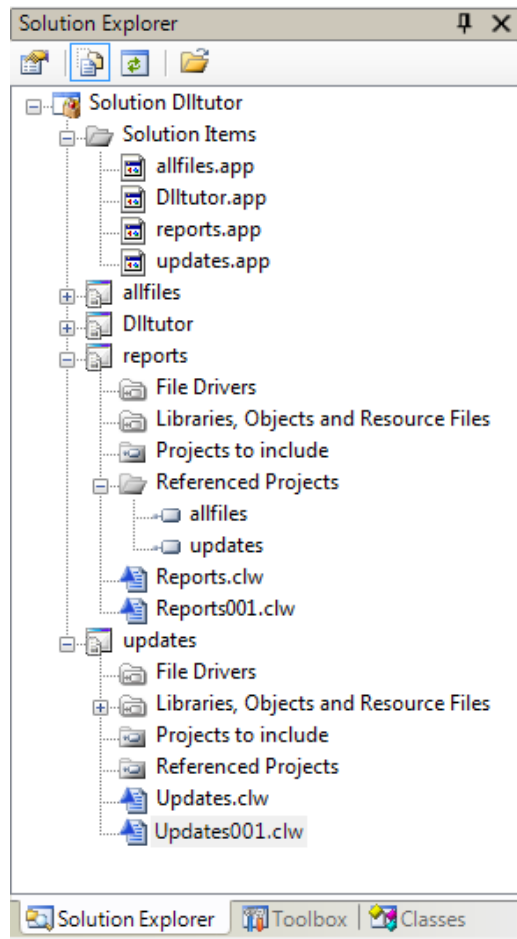


Figure 1. The Solution Explorer

Similarly, the file open dialog option now offers you an option for Solution, Project or Application, which correspond to .sln, .cwproj, .pr, .prj or .app files. Open a .pr or .prj file and the IDE will create a new solution file (.sln) and a new project file (.cwproj). At that point you can delete (or ignore) the original .pr or .prj file.

Open an .app file and the IDE will (after prompting you) convert the .app file in-place to the new format and create a new solution file containing the .app and a newly created .cwproj file which contains the generated source. For more on how all this works see [Understanding The C7 Build System](#).

A number of new template symbols have been introduced to make it easier to deal with prefixed local and global data.

There are new variants on GetPosition/SetPosition functions which let you set/get the position of an APPLICATION, WINDOW or REPORT that is *not* the topmost structure. The additional parameter is the label for or reference to that structure.

New stuff not yet done

Apparently work is being done on new save functionality, such that you will be able to save your changes in the embeditor and generate/compile *without* having to leave the embeditor. I hope that's extended to other parts of the AppGen. It would be a great time saver, and would more than make up for the current lack of keyboard shortcuts (see below).

Interesting stuff

Both C6 and C7 provide a text view of the window structure which you can edit. Sometimes it's far easier to make a few quick changes in the text version than to edit the visual presentation (as when you want to change the order of fields or do a mass revision of an entry picture).

In C6, if you somehow mung up the window structure you'll get this message:

```
Error: Syntax error reading WINDOW or REPORT
```

That's not particularly helpful. You'll be taken back to the editor and you'll have the option of making further changes or losing all your changes. That can be a big pain if you've made a lot of changes.

C7 provides a more meaningful error message; here's a truncated example that shows what happens if I misspell PROMPT as PROMP:

```
QuickWindow WINDOW('Form Titleauthor'),AT(,163,98),|
  FONT('MS Sans Serif',8,,CHARSET:DEFAULT), |
  RESIZE,CENTER,GRAY,IMM,MDI,HLP('UpdateTitleauthor'),SYSTEM
  SHEET,AT(4,4,155,72),USE(?CurrentTab),#ORIG(CurrentTab),#ORDINAL(1)
  TAB('&1 General'),USE(?Tab:1),#ORDINAL(2)
  PROMP('Authors SS:'),AT(8,20),USE(?TTA:Au_SS:Prompt)|
!!> ERROR(15): Unknown keyword: PROMP
      ,TRN,#LINK(TTA:Au_SS),#ORIG(?TTA:Au_SS:Prompt),#ORDINAL(3)
  END
  END
  END
```

```
!!> ERROR: Unexpected text after closing END
```

Besides fixing the problem you need to manually delete the error message.

One thing of note: this isn't a standard text editing window, it's more like a text box with custom error handling. Hopefully this will be changed to a regular text editor with full search/replace, undo etc. Undo especially. But most developers probably never use this feature anyway, so I don't imagine a major overhaul is high on the priority list.

Usability

Shortcuts, shortcuts, shortcuts! That's the rallying cry. Yes, there will be more keyboard shortcuts as the AppGen is finalized. Right now there's a fair bit of mouse clicking needed to get around, particularly when it comes to saving changes/closing windows, but SV is making all the right noises about addressing usability issues.

It's not all down to shortcuts, however. There are some areas where workflow has changed and developers will need to get used to doing things in a slightly different way. This is particularly true of the Properties pad.

Properties pad

One of the biggest changes between C6 and C7 is the way control properties are handled. Take entry controls: in C6 you right-click on a control to see its property dialog, as in Figure 2.

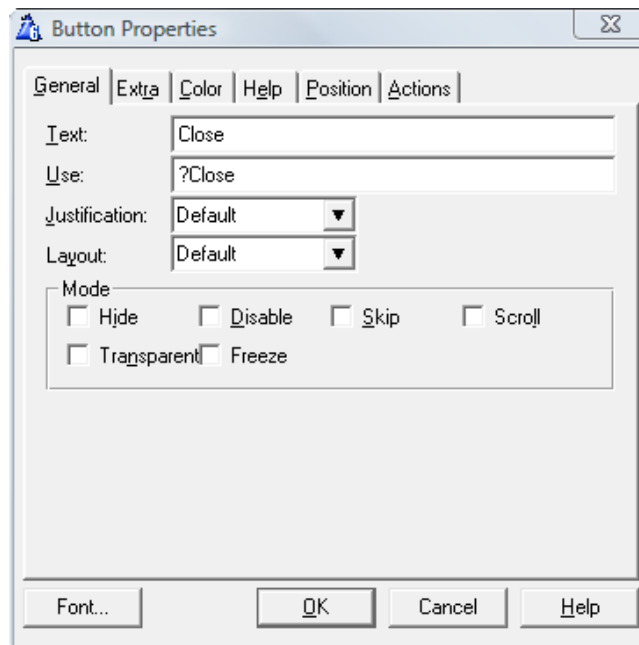


Figure 2. The C6 property dialog

In C7 the information on this dialog is displayed in the Properties pad using a PropertyGrid control (and yes, you can use these in your own Clarion# apps if you wish). In Figure 3 I've clicked on a Close button the window designer.

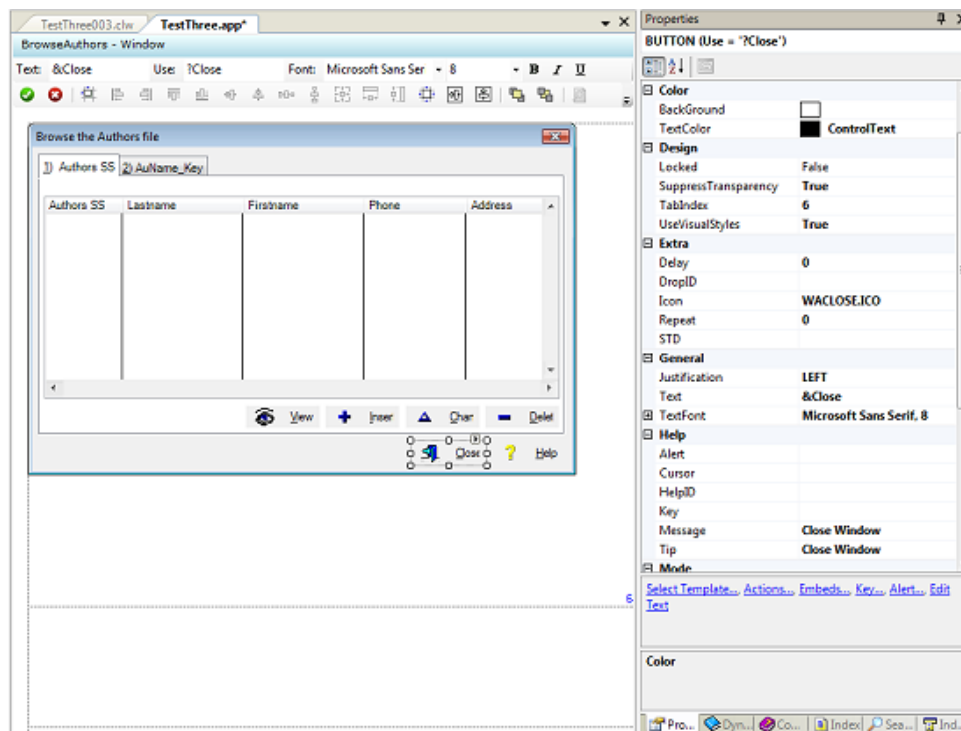


Figure 3. Selecting a control in the window designer ([view larger image](#))

Figure 4 shows a close-up of the Properties pad. Now, rather than navigating from tab to tab you move up and down the Properties pad using the scroll bar. You can also collapse and expand categories as you wish, and the IDE will remember your settings until you next start up the IDE. You can move the Properties pad all around the IDE, or even have it

in its own window outside the IDE. This is definitely a more flexible, less modal way to work with properties.

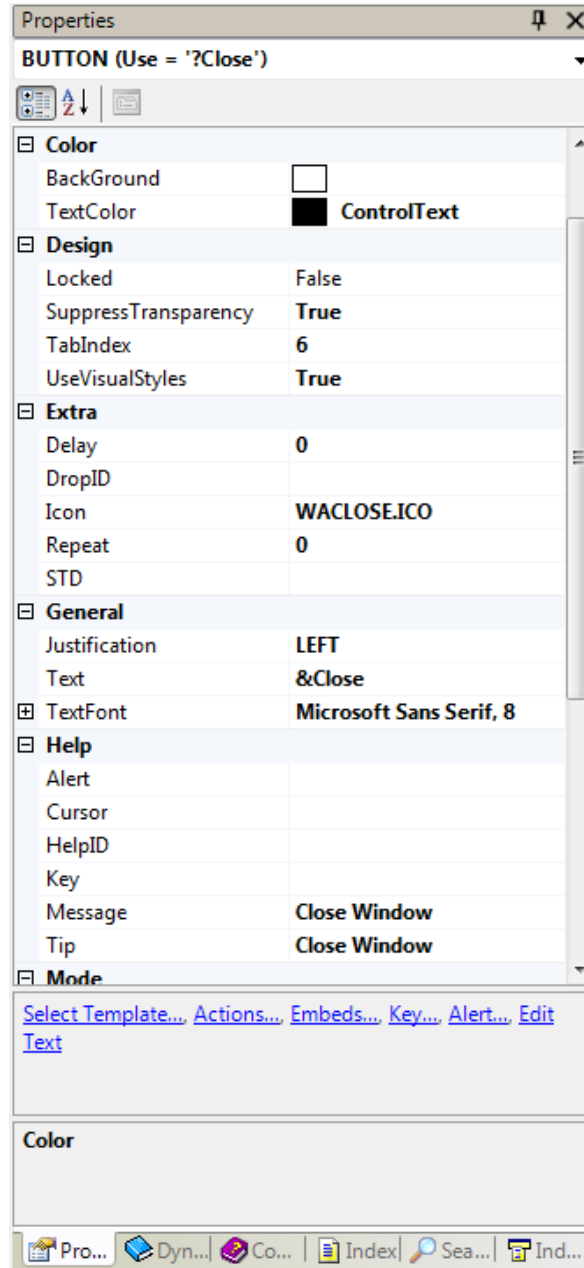


Figure 4. The Properties pad (PropertyGrid control)

But not everyone is completely happy with the PropertyGrid approach, MS standard though it may be. For one thing, True/False settings initially seem more complex than they should be. Instead of clicking on a checkbox, you click on the value (True or False) and a two-item droplist appears, letting you select the desired value (True or False). That seems like a lot more work than a checkbox. You can shorten the process a little by double-clicking on the value, which bypasses the droplist and simply toggles True to False and vice versa. Better, but still not as visually helpful as a checkbox. Bob Z has said SV will look into options for improving the visuals.

Another issue is that the arrangement of items in the C6 dialogs isn't exactly mimicked by the C7 property list. And while you can toggle between group view and alphabetical view, alphabetical view usually isn't all that useful. There is an inactive Property Pages button (a sort button?) in addition to buttons for group and alphabetical views, but it's, well, inactive.

Some things are still modal. If you click on an entry field in the window designer you'll see a little popup menu indicator sitting next to the upper right-hand selector. This is a way to get to additional tasks (which in most cases are also listed on the Properties pad - see Figure 4).

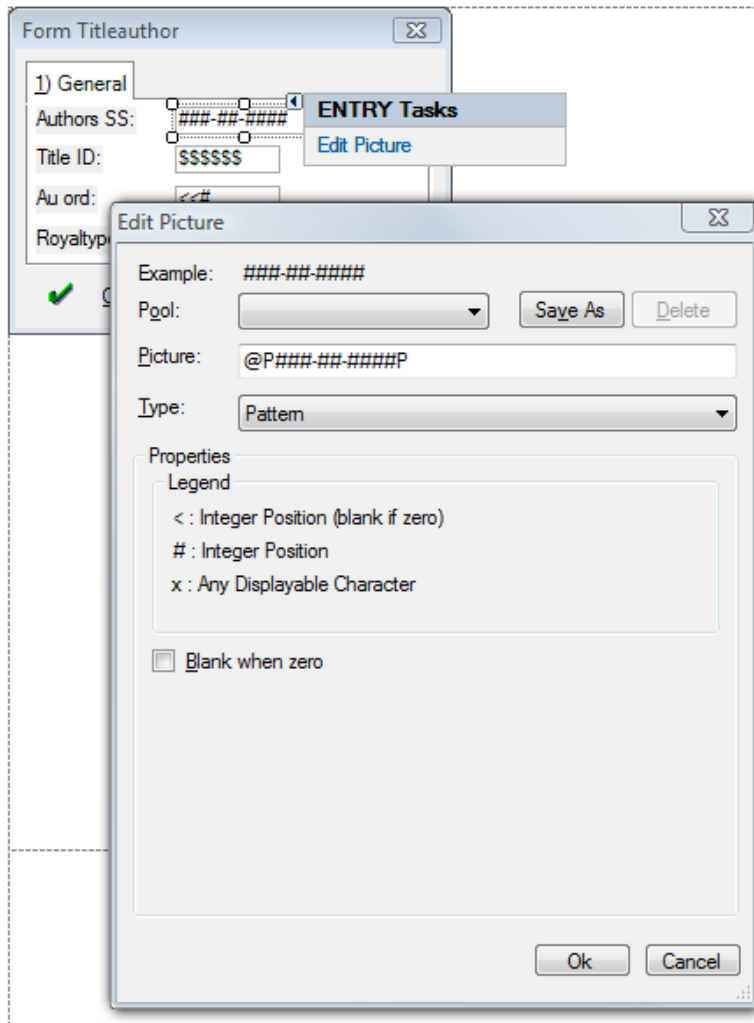


Figure 5. Editing an entry field's picture

In fact, it seems to me that most of these tasks used to be listed via that control menu, whereas now they seem to mostly be listed in the Properties pad. I don't know what the long term plan is, but in any case this points out some of the limitations of the PropertyGrid control.

Some items are available all three ways. You can edit an entry field's picture via the control menu, via the link at the bottom of the Properties pad, or by editing the Properties value directly (and there's also a button there to bring up the picture dialog).

At present there's no way to do a mass edit of controls as you can in C6 (press F12 in the window formatter). Hopefully that's another feature that will be ported to C7.

Conclusions

It's been interesting watching (and participating in) the dialog between SV and the third party folks. As I said earlier, the focus seems to have shifted away from critical bugs to usability. One of the common patterns of communication goes something like this:

Us: Why can't I do task X as easily in C7 as in C6?

SV: Well, we've changed that. You now do it this way.

Us: That way takes four steps; I'm used to doing it in two. Make it like C6.

Now, sometimes SV says "Whoops, sorry. We'll fix that for the next release." Those are the easy ones (for us, at least).

Sometimes SV doesn't agree that the new way is more work. And sometimes they're right, and the problem is that we haven't learned how make the most of the new functionality, or we don't even realize what's there. Moving the various pads around the UI is like that; more than once Bob Z or someone else has explained how you can configure your workspace so the things you need are readily available and not hidden behind some other tab or window.

Other times there's a protracted discussion about the particular task/feature (or even a bug), and what's immediately obvious to us isn't immediately obvious to SV because, like all users, we use the product in ways they haven't anticipated. They're not always convinced either, but the quality of communication is, on the whole, pretty good.

Bottom line: it's possible, under the right circumstances, to do actual work with C7 AppGen now. But would I recommend it for production work? No. Give it a few more beta releases.

I expect we'll see at least one more third party release before any wider release to CSP participants, given that some core template issues have not yet been resolved. And despite some pretty thorough testing by third party folks, it will probably take a CSP release or three to shake out the last showstoppers.

Addendum

A couple of late-breaking notes. I've heard that all the shortcut keys in the C6 menu editor have now been implemented for C7, along with a few other shortcuts. And the property tree is now auto-expanded, making it easier to do things like adjust window position.

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 Wednesday, November 26, 2008 by Michael Gorman

Dave:

Thanks for the review. Hearing it from SV is Golden, but hearing it from you is Goldener. Working with it myself hopefully will be Goldenest.

Regards,

Mike Gorman

[Add a comment](#)

Clarion Magazine

Dual or Quad Core For Clarion?

by Dave Harms

Published 2008-11-17

For the longest time a personal computer's CPU chip contained just one execution core. In other words, a CPU was just a CPU. Then in 2001 IBM released the POWER4, the first microprocessor to contain two execution cores - essentially two processors in one. In 2004 Intel demonstrated its first dual-core Pentium, which began shipping in 2005.

These days dual core CPUs are pretty much the minimum processor for new midrange systems. But what about quad core? Does having four cores in a CPU offer any tangible benefit to Clarion developers?

I have a little story to tell that may offer some insight. But please keep in mind that this isn't a guide to building high performance systems, more a brief study of one of the key components. There are many performance factors and the CPU is just one, even if it's a big one.

A tale of woe

I recently decided it was time to update my somewhat antiquated Athlon 64/Vista 32 development machine, but with minimal disruption to the existing system. Replacing the motherboard typically means repairing Windows, which is not without risk. And I didn't feel up to a complete Windows reinstall. My initial thought was to drop in a faster CPU (an Opteron) as a stopgap. And before I did that I figured I'd better flash the BIOS.

Can you hear the ominous background music?

I've flashed BIOSes on other hardware, and it's away's worked fine. But I'm never comfortable with the process. So I made sure I had good backups (although presumably a bad BIOS wasn't going to toast my RAID 1 array) and downloaded the [Ultimate Boot CD ISO](#). I then set up [UBCD on a USB flash drive](#). I copied the Asus flash utility and the new BIOS to the USB drive and rebooted. (I had to go into BIOS setup and specify the USB drive as a boot device; on this board (A8N-SLI) that option is only present when a bootable USB device is attached.)

Once I'd booted to the flash drive I ran a DOS shell and updated the BIOS. But when I rebooted, the BIOS said it couldn't find the boot CD. Would I be so kind as to insert the disk and press a key? That didn't seem right - I wanted to boot from the hard drive. After a couple of reboots yielded the same message I decided to turn off the power.

Big mistake.

When I turned the power back on the motherboard wouldn't even do a power on self-test (POST). Not a peep. Nothing.

Well, it was a Saturday and I had lots of other things to do, so I decided to just let the computer sit with the power off for a while. But every time I came back to the machine the result was the same: nothing. No POST at all.

Some Internet searching failed to turn up an obvious answer. So I figured I might as well bite the bullet and upgrade the motherboard.

Despite this particular experience, I'm generally a fan of Asus boards, and after some research I settled on the P5Q which supports both Core 2 (dual core) and Core 2 Duo (quad core) CPUs. And that, of course, meant another choice: how many cores did I want?

Cores vs processor speed

You don't see a lot of single-core processors around anymore; just about everything in the stores these days is dual or quad core. So why bother with dual core - isn't quad core always better?

The answer, unsurprisingly, is "it depends."

For Clarion development, dual core processors, at least on the surface, offer more bang for the buck. That's because at a given price point the dual core CPU will have a higher processor speed, and for single-threaded tasks such as code generation and compiling there's no benefit to having additional cores. (Actually, code generation probably *could* be made multi-threaded. I don't think there's any technical reason why C7 couldn't generate, say, four apps at once on a quad core box, given that each app already gets its own copy of the generation environment. Parallel compiles, however, are trickier because dependencies have to be resolved.)

My local retailer is selling the Intel Core 2 (dual core) E8400 CPU for about \$10 less than the Intel Core 2 Quad Q6600. The E8400 runs at 3 GHz, the Q6600 at 2.4 GHz. That's a 25% boost in CPU speed for slightly less money.

So forget about quad core and get dual core, right? Well, maybe....

If you spend a lot of time actually compiling, and while you're compiling you're doing nothing productive, then a faster dual-core processor may be the answer. On the other hand, C7 already generates and compiles faster than C6, so generate/compile speed may already be sufficient with a slightly slower quad core CPU.

Typically quad core CPUs are recommended for graphics-oriented tasks such as video rendering, where applications can readily benefit from multiple threads with each thread executing on its own processor. But if you run a lot of applications at once time, and some of those applications gobble up CPU cycles, having a few extra cores around is only going to help.

I ended up with Q6600, which is hardly bleeding edge these days, and I'm amazed at how much power this processor has. No doubt the effect is more dramatic going from one core to four, but any way you look at it four cores put a lot of cycles at your disposal. While writing this article I fired up Beyond Compare to do a disk synchronization, I started rendering a video with Sony Vegas and I generated all the source for all four DLLTutor applications in C7. I had another half dozen applications sitting idle. My system remained completely responsive at all times, and C7 code generation time was essentially unchanged from what it would be if nothing else was running. I did finally manage to increase C7 generation time by about 50% after I replaced the Beyond Compare task with Visual Studio 2008 building a large C# application. Hey, maybe I need eight cores....

Somewhere in my research I came across a post comparing an overclocked dual core CPU to a high-revving V12 gas engine, and a quad to a diesel engine with monster torque. That isn't a bad analogy, although these days the quads aren't lagging all that far behind the dual core systems when it comes to processor speed.

Some (sort of) real world numbers

As I mentioned, my desktop is now running a Core 2 Quad Q6600 at 2.4GHz (I haven't tried overclocking yet) with Vista 32, and my laptop is running a Core 2 Duo at 2.1 GHz with Vista 64. Both systems have 4 GB RAM (although only Vista 64 can actually use all that memory). The laptop has a 5400 RPM drive, the desktop a 7200 RPM drive. The Q6600 has 33% more cache per core. The time to generate the DLLTutor apps is almost identical on both machines, coming in at 13 seconds for the quad core and 14 seconds for the dual core.

While it's impossible to be precise given the hardware differences, I think this illustrates the importance of processor speed. For single-threaded tasks like compiling (and assuming there aren't other loads on the system) there's no denying the appeal of dual core.

Power consumption

One other point worth keeping in mind is power consumption. More cores = more power, and even if you're not concerned with your energy bill, more power means more heat needs to be removed from the CPU. I have a honking big [Ninja cooler](#) on mine and monitor temps with [SpeedFan](#); I seldom see any of the cores reach 40C in normal usage.

At max load the Q6600 reportedly uses about 50% more power than a similar dual core (~100 watts, more if overclocked); at idle, which is what most of our processors do most of the time, the difference is probably much less. Going by [this chart](#) an E6600 has the same standby usage as the Q6600; that suggests that over the course of a typical programming day, with the majority of time spent editing and doing other non-processor-intensive tasks that overall power use isn't going to be significantly higher with a quad core.

Power consumption also varies by the manufacturer and the technology used to create the chip. For instance, Intel's newer 45nm process chips are more energy efficient than the 65nm process chips.

Summary

There are a lot of factors to consider when evaluating performance, and while CPU is only one of them it's clearly one of the most important. For dedicated Clarion development (at least as we now know it) a quad core CPU is most likely overkill, and more expensive as well for a given clock speed.

If, however, you find yourself running many applications at the same time as you run Clarion, and some of those applications are processor-intensive, you may find the extra cost and/or a slightly slower clock speed associated with a quad-core CPU a welcome trade-off.

But if you want a high-performance dev machine...

As I said earlier, this isn't an article on building performance systems, just a story about CPUs. But feel free to add your suggestions in the comments. I know of a few developers who have bought gaming systems on the theory that if it's fast enough to keep a gamer happy, it's fast enough to keep a programmer happy. And that's a pretty good place to start.

Let me add two more concepts to consider: noise and data security.

I like quiet systems, and quiet and fast don't always go together. Bigger, slower fans can move as much air as smaller, faster fans, and sometimes the tiny fans make the most noise. Passive heat sinks are of course completely silent, although they too usually require some airflow. Variable speed fans and temperature monitoring software let you balance noise and airflow. And some hard drives are quieter than others (although just about anything you buy these days is a vast improvement over the screaming meemies of yesteryear).

Speaking of drives, for a dev machine I like them paired in a RAID 1 mirror. You can find RAID support on lots of motherboards. In fact I have a problem at the moment: I can't get Vista to successfully boot when I switch my new motherboard to RAID mode (yes, I've installed the RAID driver), so I'm stuck using one of my former RAIDed drives and periodically creating an image with [Acronis](#). Looks I'll have to do that Windows reinstall anyway....

[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 Tuesday, November 18, 2008 by Richard Rose

One thing to consider is that motherboards now have built in intelligence to reduce power consumption even further. In your P5Q motherboards case its called AI NAP.

Posted on Tuesday, November 18, 2008 by Jorge Alejandro Lavera

Hi, David.

I think multiples cores can have major importance when you use Virtual machines. For example, I tied each VM to a single core, and now I often compile using C6 on two projects at the same time (each one in a separate VM) on a dual core Cpu. The cpu goes to 100% the whole time (i.e. both cores are really in full use), and the machine still is responsive for other tasks... I guess if I'd have four cores I could generate and compile four projects in the same total time!

Posted on Wednesday, November 19, 2008 by Dave Harms

Jorge,

Good point!

Dave

[Add a comment](#)

Clarion Magazine

Prompting For Time

by Paul Blais

Published 2008-11-13

When I prompt users for information I try to find a context to help them supply the data easily and accurately. If I prompt for a date I provide a calendar picker. I use a color picker for a screen color. But there's no Clarion supplied picker for the time of day, and few solutions come to mind.

So what would a time picker look like? One context for time is the sun; I can represent a day graphically as a gradient bitmap from midnight to midnight. I can visually find the time desired, and the change from dark to light tends to draw the eye toward the center of the control.

In fact, what I typically need is not just to pick a point in time, but a block of time on any given date. In other words, I want an "appointment picker" such as the one shown in Figure 1.

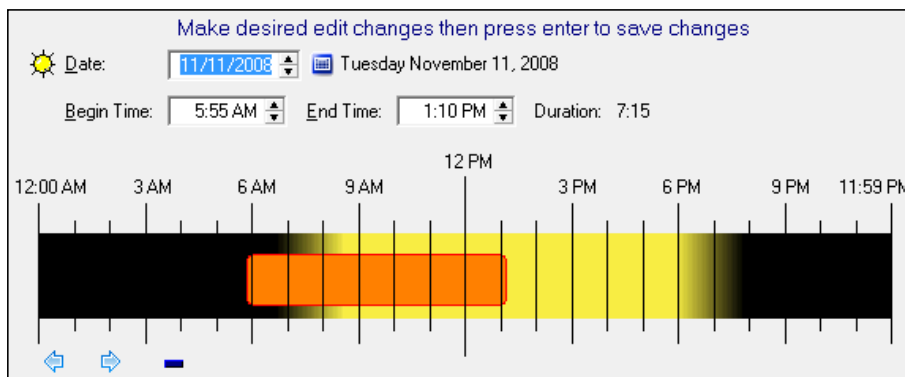


Figure 1. The appointment picker control

I will describe aspects of the appointment picker source code in the following order: first, I'll cover the basics of using the sunrise /sunset computation; second, I will explore the graphic representation of a date relative to the sun; and third, I'll look at the actual picker procedure. For the full source code please refer to the accompanying download.

Solar computations

The angle of the sun relative to the horizon is what divides day and night; when the sun angle is near and yet below the horizon the result is twilight. This is a dynamic relationship based on the earth and sun position relative to the earth's rotation and the orientation of the earth's axis which changes throughout the four seasons.

The computation for every location on the planet gets complicated in the higher latitudes, however. On dates at some locations the sun never fully sets, or fails to rise or can rise and set more than once per day. I didn't invest enough time to do all the iterative mathematics, so the range of the function supplied is limited to about 55 degrees north and south latitude at any longitude on any date. Refinement requires additional iterative computations for the higher latitudes. In the mid and lower latitudes the accuracy is plus or minus one minute or two.

The computation requires the date, location latitude, location longitude, the UTC time zone offset and a flag indicating whether to observe US Daylight Savings time. Locations outside the US with different daylight savings rules can be easily accommodated in the routine that computes an aggregate time offset based on the UTC offset and the daylight savings rule based on the date.

All of these computations are handled by the Src:SunTimes procedure, which you can find in the downloaded source.

I create a GROUP structure and label it as a defined Type. I can now pass the group by reference to the Src:SunTimes procedure which is totally thread safe as it only uses local data and the passed group.

```
tSunTimes      GROUP,TYPE,PRE()
Date           LONG      ! clarion date
Latitude       REAL      ! decimal value + = N and - = S
Longitude      REAL      ! decimal value + = E and - = S
UTCcorrect     LONG      ! UTC time zone offset
SunButton      USHORT    ! Sun button to receive tooltip (optional)
USDaylightSav  BYTE      ! 1 = observe daylight savings
R1            LONG      ! Official sunrise
R2            LONG      ! Civil sunrise
R3            LONG      ! Nautical sunrise
R4            LONG      ! Astronomical sunrise
S1            LONG      ! Official sunset
S2            LONG      ! Civil sunset
S3            LONG      ! Nautical sunset
S4            LONG      ! Astronomical sunset
END
```

I return the results in the same Group, and any invalid results come back as zero. I refuse a computation if the requisite parameters are blank or zero.

I compute four sun angles for day and for night. The following equates expressed as decimal degrees represent angles of the sun to the horizon where zero is directly vertical:

```
zOffical      REAL(90.833333333333)
zCivil        REAL(96.0)
zNautical     REAL(102.0)
zAstro        REAL(108.0)
```

Official sunrise and sunset appear when the upper rim of the sun appears on the horizon. The earth's atmosphere bends the light such that this occurs 0.8333 degrees below the horizon, which means that when I see the sun set it already set seconds ago.

Civil twilight is the time when I can no longer read a newspaper when the sun is 6 degrees below the horizon. Nautical twilight is when I can no longer see the horizon through a sextant when the sun is 12 degrees below the horizon.

Astronomical sunset is when the sky is as dark as it can be, with the sun at 18 degrees below the horizon. The bending of light by the atmosphere makes the daylight last longer and sunrise appear sooner, unlike the moon which has no atmosphere therefore no twilight.

The time between official sunset and astronomical sunset varies greatly based on the latitude and the season. My control indicates this twilight period as a gradient shade stretched from astronomical sunrise to official sunrise and then from official sunset to astronomical sunset. I can compute those times of the year in the higher latitudes when there is no astronomical sunrise or sunset, but at higher latitudes the error increases a lot.

See the Src:SunTimes procedure for all the code required. The group can optionally include a button control use variable that will have a dynamic tool tip created and assigned. Clicking the button will capture the tool tip information of the sun times to the clipboard. A sun icon button is included to add this feature to any date display I like.

Setting up the demonstration application

The Main screen has a setup tab for parameters and a simulated entry form for an appointment. I begin with the second tab to set up location and time parameters unique to me. The function will not operate at all unless these values are supplied. I use Google Earth to find my own location parameters with reasonable accuracy - you can also try Satellite Signals [Lat/Long finder page](#). West longitude and south latitude are by convention negative values. Another way to obtain location information is via a zip code/postal code database.

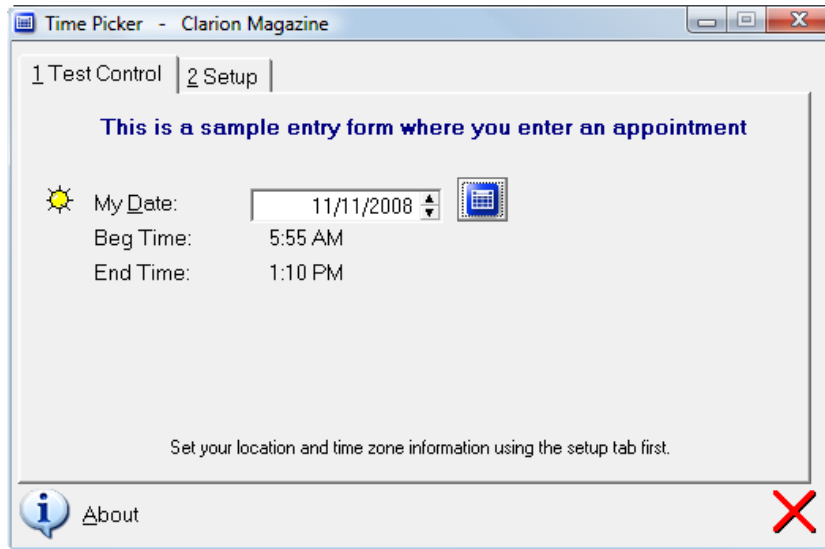


Figure 2. Demo program, main tab

Note the format of the numbers on the setup tab. I could use the Windows API to derive the needed time zone information but I want the ability to work with any appointment in any location, not just with reference to my own location.

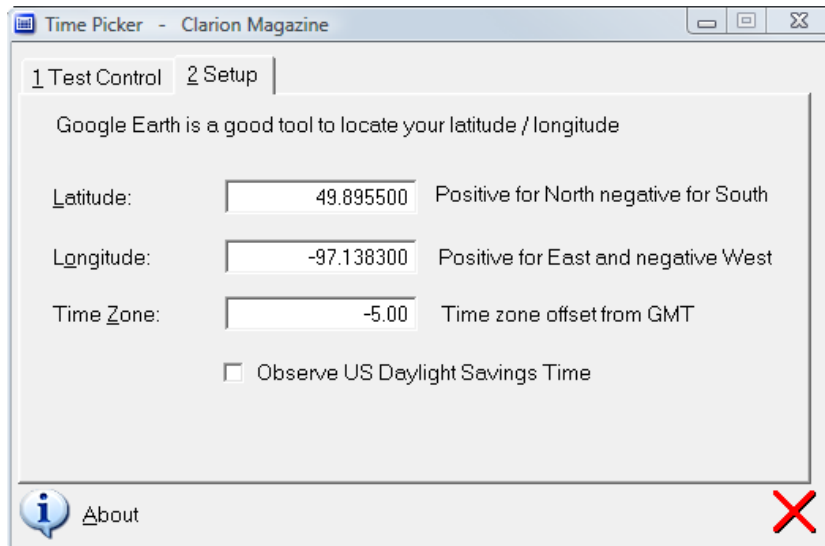


Figure 3. Demo program, Setup tab

The entry form simulation screen on tab 1 has a spin button field for a date and displays two times. A sun button and a date picker button are included too. From this main screen I provide sun time information as a tool tip on the sun button. I don't want to clutter the screen with information so the tool tip will allow easy access without distraction. I can change the date and hover over the button to see it, and click on the button to capture the screen tool tip text to the clipboard.

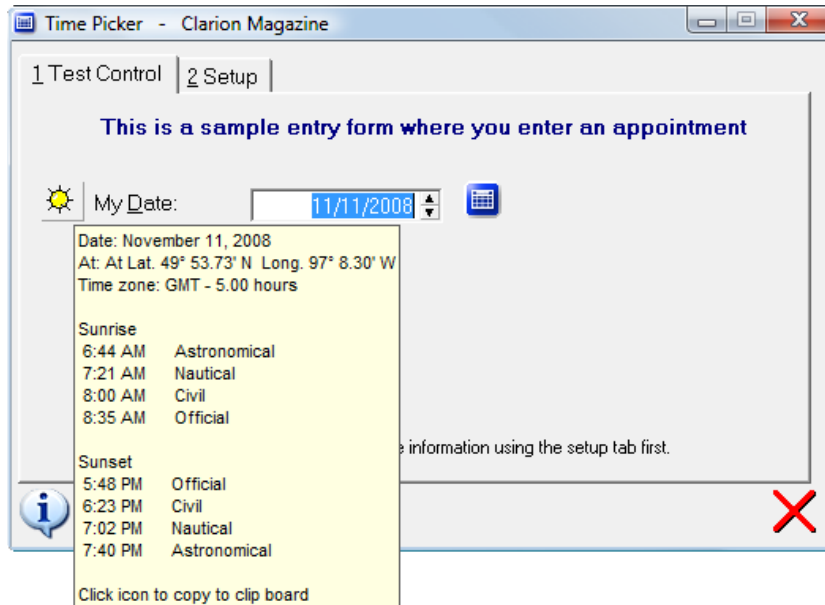


Figure 4. Displaying the sun times tooltip

I added the following code to the ThisWindow.Reset embed point of the main screen. The procedure Src:SunTimes recomputes each time the date changes. I force a ThisWindow.Reset on the EventAccepted and NewSelection embed points for the date entry field. NewSelection events are triggered by the spin button.

```

IF MyDate > 0 AND GLO:Latitude AND GLO:Longitude
  ?Sun{Prop:Hide} = False
  ! Init our Suntimes group from the saved global settings
  ! reset every time in case the user changes them
  SunTimes.Latitude = GLO:Latitude
  SunTimes.Longitude = GLO:Longitude
  SunTimes.UTCcorrect = GLO:Timezone
  SunTimes.USDaylightSav = GLO:USDaylightSav
  SunTimes.SunButton = ?Sun
  SunTimes.Date = MyDate
  Src:SunTimes(SunTimes) ! Compute the sun times
ELSE
  ?Sun{Prop:Hide} = True
END
    
```

The global values save to an INI file for this demo. Using XML would make this even easier. I could also look up the global values on the fly, say for different entries in a contacts database.

Making a time picker

Making the picker requires a window procedure. I pass the main screen values by reference to the picker and the picker will alter or not alter the values as desired by the user. I prompt for the date and times on the same procedure. I use a standard Clarion Date picker for the example.

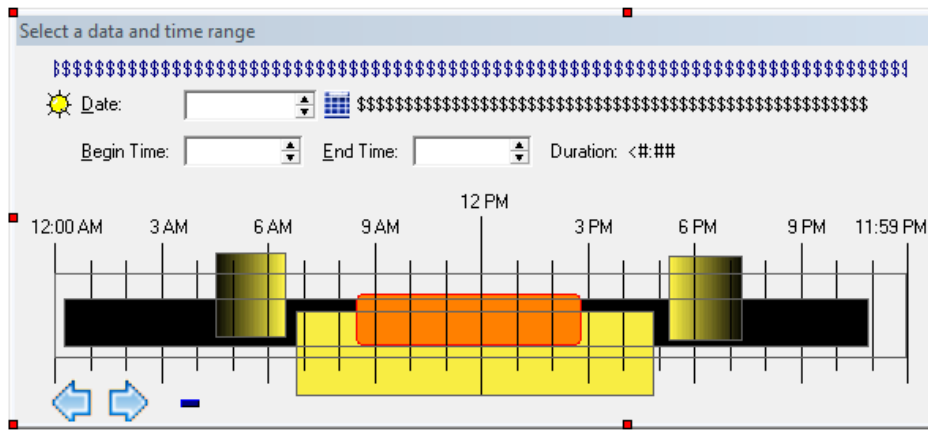


Figure 5. The Time Picker in the IDE

The above is a screen capture from the Clarion IDE showing the picker screen. There are local variables for the date, begin time, and end time. I added a Clarion date picker next to the date and to the right of the date I display the date's text value complete with day-of-the-week name. As on the entry form I added a sun button with detailed information in its tooltip. Below the entry fields I created the time picker control.

Clarion screen structures allow layering using a last declared is on top rule. Here is the order of the screen elements you see:

1. A hidden rectangular Region control called ?Region1 with the IMM attribute. This traps mouse events only inside its boundary.
2. An image control using night.jpg called ?Night is solid black. I could make one with stars, a moon, and planets.
3. On top of ?Night I lay out three more images: ?Sunrise, ?Day, and ?Sunset. They can be any place but should be located in the screen structure after the ?Night image control. Two twilight bitmaps are included in the source for those latitudes that don't have total darkness at night. They are added to the project as resources but do not require an image control.
4. A rectangular box with round attribute filled with a color I like. This represents the user defined time period that will grow and be relocated in real time. The goal of the whole procedure is to display a time range on top of the sunrise to sunset images.
5. The last layer includes all the vertical lines spaced evenly with text labels; the lines overlay the bitmaps. In the AppGen I used the ellipsis button on the procedure properties screen next to the Window button to set the order exactly as I've described it. I find it often is easier to do this work by hand. I also make all Y and X properties of the lines exactly as I require by hand code.

The code that moves and resizes the bitmaps is located in the Src:SetDaylight procedure. (I isolate this code because I also use the same approach for a header above a browse box where the columns denote half hour time increments - that code is not included with this article.) The redraw code is triggered by the NewSelection event of the date spin buttons and by the popup date picker. Times computed from the Suntimes group are passed to the Src:SetDaylight code to stretch the bitmaps as required to give the illusion of one bitmap when there really are four bitmaps edge matched.

Under the bitmaps I added button arrows left and right that increment or decrement the time period by five minute increments, thus shifting the period forward or back without changing the period's length. The delete button removes both times. I added hot keys to the time fields and set the spin button increments to five minutes or 30000 or Clarion time format. A hot key to the time field lets you use the arrow up and down key (or mouse wheel) to edit the end point in five minute increments. This provides all the elements for keyboard or mouse based control.

The picker process

I broke down the picker process into steps by first defining what the user can do and how I choose to let them operate. This is a clear case where writing the documentation before I write the code helps design the code.

I already know how the keyboard works so the following is how the mouse operates:

1. On a screen with no times I show an I-beam cursor when the mouse moves over the time picker.
2. After the first MouseLeft click I save the begin time and init an end time period of plus five minutes. Keeping the button depressed I can drag the mouse left and right and cause the end time to be changed dynamically as I drag. When I release the mouse the end time is finalized and the drag operation is completed.
3. Once a time range has been established the cursor shape changes to indicate the time period can be edited. If my cursor is inside ?Region1 I display the cursor as a normal pointer cursor. If my cursor is near an end of the rectangle time range the cursor is an east/west cursor. If my cursor is inside the rectangle and not near an end I display a move cursor. I know that when I am seeing those cursor shapes I can operate a mouse function to perform that operation.
4. To activate a mouse function I depress the button and leave it depressed. This will arm a drag function either from an end point or the entire range based on the type of cursor displayed.
5. If I continue to press and move the cursor (dragging) to the left or right the values for the begin time and end time are adjusted dynamically. The adjustment is in five minute increments and affects either start time or end time if an end point is being dragged, or both if the range is being dragged.
6. When I release the mouse the drag operation terminates and the cursor shape reverts to normal.
7. All through this process I display a message line at the top of the screen and I update the display of the rectangle as well as the entry fields. The granularity of the events makes it appear as a fluid motion. The time period rectangle grows and shrinks like a rubber band.

Capturing the time range

The picker for the time period traps Mouse up events inside the ?Region1 control; MouseX() and MouseY() return the actual X and Y values in screen coordinates when the event happened. Code will execute for each event and no other code will run before the processing completes. The hand is not quicker than the computer. Events processed quickly look like fluid motion.

Event:MouseIn and Event:MouseOut are posted when the mouse moves in to or out of the ?Region1 control. Button:Down and Button:Up are posted when any mouse button is depressed or released inside ?Region1. Event:Move is posted any time the mouse changes its X and Y location inside ?Region1.

The mouse code is in two routines.in the time picker procedure. The ComputeMouse routine converts a mouse X location into a time. It also keeps the state conditions of the cursor type and the operation type as well as the state of the details of the active operation. ComputeMouse will also assign time values from the mouse. The ComputeActive routine converts the begin and end time into a rectangle location and width and displays the appropriate message to the user, based on the cursor and drag states. ComputeActive is in a separate routine because the entire process can be driven from the keyboard or the mouse with the same result.

The ComputeMouse routine is called only from the ThisWindow.TakeFieldEvent embed:

```
ThisWindow.TakeFieldEvent PROCEDURE
```

```
ReturnValu     BYTE,AUTO
```

```
Looped BYTE
```

```
CODE
```

```
LOOP
```

```
IF Looped
```

```
    RETURN Level:Notify
```

```
ELSE
```

```
    Looped = 1
```

```
END
```

```
CASE FIELD()
```

```

OF ?Region1
DO ComputeMouse ! all the event code is in one routine
END
    
```

One small tweak is required to prevent an undesirable side effect. I add a local a true / false byte variable called BlockMouse. I set the flag to true before the call to the Clarion date picker and then reset false upon returning. Should the date picker overlay the time region, ?Region1 will post the mouse events from the top level date picker, which can make a mess of the users screen. I allow the ComputeMouse and ComputeActive routines to exit if the block is set to true. Having only one point in the code where ComputeMouse is called makes all of this easier to manage.

The two compute routines inside the picker procedure rely on Clarion runtime properties and events. The picker is set up for four minutes of clock time per screen unit of resolution returned from the XMouse() Clarion runtime property. This allows selection of times within five minutes for a trained mouse user.

I added a few other refinements after testing. I control the cursor for end point dragging a little differently than I do for range dragging. It's really hard to "grab" the end point exactly within one pixel. I added an extra "dragzone" of one more screen unit on either side of the rectangle border. This provides a bigger target at which to aim a mouse. Selecting the inside of the region is easy so no adjustment is required. I allow the end point to be dragged across the full range of the time scale. When a begin time is dragged to the right of the end time I detect the situation and swap the two end point values as well as change the drag mode from a begin point drag to an end point drag. This gives the illusion of a single point nailed to the screen and the other end totally flexible. To further identify the drag mode the fill color of the rectangle changes to transparent when the rectangle is being dragged in any drag mode.

When beginning on a procedure window that has no interval set there is no rectangle displayed. If *either* end point is blank the "I beam" cursor displays and can be used to reset the start point. Every initialization creates a default end point of plus 5 minutes, then switches to end point drag mode.

I have also eliminated the pesky problem of no range but only one end point. Doing the dynamic swapping of end points prevents backwards time periods as well as simplifies the code. A user should be able to click at some point then drag left to right or right to left.

The demo application includes some debug information on the screen. It can prove a good tool to study the inner workings and to try other expanded functionality.

Figure 6 shows the completed appointment picker:

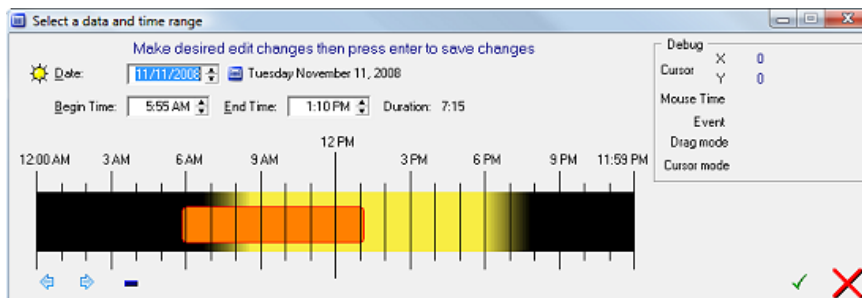


Figure 6. TimePick1.png (view full size image)

Possible improvements

You can switch to a coarser time resolution through a change in the graphics and by changing the value of the eScreenResolution EQUATE(). My approach rounds to a reasonable level of detail yet permits times to the exact minute with manual adjustment. To make a picker with one minute resolution requires a screen resolution far too high for common desktop systems, but I can imagine a magnification feature for precision time selection. It would not be hard to add that feature.

Using run time properties I can move the bit maps and force them to create the illusion of a single bitmap. Keyboard controls allow easy adjustments when modification is required and for non mouse users. Providing the date and time

pickers together means all the tools for time are on one screen and that screen can be reused any place in the application. With a small modification I could change the picker to be a selection of one time. This picker would allow equal begin and end times.

Summary

The mouse and the use of runtime events make all sorts of dynamic GUI designs possible. I am in the hunt for code to compute moon phases. It would be nice to have little tiny moons display at the proper moon rise and moon set times. Interesting little features do make for a more engaging application, and prompting for user information in context makes data entry faster and more accurate.

[Download the source](#)

[Paul Blais](#) has been a Clarion developer since CPD version 2003. He became a full time independent Clarion Developer in 1998. In 2000 he merged his life and business with his wife Barabra. The merger yielded 3 dogs, one horse, 3 vehicles, and Organizational Development Strategies, Inc.. When not writing code he and his wife can often be seen aboard Bright Eyes sailing the Chesapeake Bay.



Reader Comments

Posted on Monday, November 17, 2008 by Stephen Ryan

wonderful !

Posted on Wednesday, November 19, 2008 by Friedrich Linder

Very interesting!!!!

Friedrich

[Add a comment](#)