

Clarion Magazine

[DevCon 2004: Clarion ASP.NET](#) SO

James Cooke reports on SoftVelocity's Clarion ASP.NET presentation.

Posted Friday, November 05, 2004

[Using Clarion's Business Rules, Part 3](#) SO

Nardus Swanevelder concludes this series by demonstrating how to implement dynamic business rules.

Posted Friday, November 05, 2004

[A Configurable Expression Editor, Part 1](#) MS

Since Clarion does not have a good integral tool for defining an expression at runtime, Tim Phillips went ahead and built one, using his Basic Editor as the foundation. Part 1 of 2.

Posted Friday, November 05, 2004

[A Facelift, A Podcast, And More Articles](#)

It's been a busy week at Clarion Magazine - we've launched the latest redesign of the web site, which was prompted by the addition of the Planet Clarion podcast, a regular audio program for Clarion developers, hosted by Dave Harms and Andrew Guidroz II. You can download the podcast the same way you download any other file, or you can use any one of a number of RSS readers to automatically download the podcast as an enclosure. Anyone can download the entire podcast; subscribers have the option of downloading individual tracks. All of the changes needed to support the podcast mean our publication schedule has been pushed back a little. Also please note that Clarion Magazine's subscription rates will be going up Dec 1, so if you're thinking of subscribing or renewing, now is the time!

Posted Wednesday, November 17, 2004

A Configurable Expression Editor, Part 2 SO

Since Clarion does not have a good integral tool for defining an expression at runtime, Tim Phillips went ahead and built one, using his Basic Editor as the foundation. In this second of two parts Tim shows how the code works.

Posted Friday, November 19, 2004

Managing Report Page Breaks With The C6 Break Manager SO

Geoff Bomford shows how to use the new BreakManagerClass to get control over report page breaks. This comprehensive article is a great introduction to this important new technology.

Posted Friday, November 26, 2004

Tackling Global Queues in Clarion 6 SO

Clarion 6 has introduced a new threading model which has definite benefits, but problems that were once easily solved with a global memory queue in earlier preemptive model cause problems in the new cooperative threading model. Many articles have been dedicated to this subject, but no general solution has appeared. Until now. John Seganakis introduces a queue management class that works smoothly with your existing embed code.

Posted Friday, November 26, 2004

Clarion/PHP Released

SoftVelocity's Clarion/PHP has been released. This product is equal in functionality to Clarion/ASP except it generates PHP code instead of ASP code, and allows

deployment to any platform supported by PHP and Apache. If you already own Clarion/ASP a special crossgrade is available.

Posted Monday, November 29, 2004

Looking for more? Check out the [site index](#), or [search the back issues](#). This site now contains more than 800 articles and a total of over a million words of Clarion-related information.

Copyright © 1999-2004 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.

You are logged in, but there is no subscription associated with this user id. To subscribe, please go to the [Clarion Magazine Store](#). If you previously had a subscription under another login, contact the [subscription department](#).

Advertisement

650 Subscriber-Only Articles

Over 950,000 Words

Try Clarion Magazine **risk free!** Take out a one year subscription (includes almost six years worth of back issues), and if within 30 days of subscribing you're not completely satisfied with Clarion Magazine, we'll refund your subscription price in full ([read the details](#)). And remember that you can cancel your subscription at any time after 30 days and receive a [prorated refund](#).

[**Subscribe Now!**](#)

[Planet Clarion for November 9, 2004](#)

In this first ever Planet Clarion podcast, hosts Dave Harms and Andrew Guidroz II discuss topics ranging from "Why stay with Clarion?" to how many developers are using Clarion, and the impact of Clarion.NET.

 [Complete MP3](#), 00:34:00, 16813 K

[Planet Clarion for November 26, 2004](#)

Planet Clarion #2 is in the can! In this edition Andrew and Dave look at color theory and application skinning, discuss the IP driver, and talk to graphic designer, Clarion developer, and rising photography star Leroy Schulz.

 [Complete MP3](#), 40:42:00, 19328 K

A full [track listing](#) is available. Paid subscribers can [download individual tracks](#).

For previous podcasts visit [the podcast page](#).

[News](#)

[Clarion Developers Challenge Week 11 Winner](#)

[Office Inside 1.58](#)

[SB5 "IP Data Server" Script](#)

[FM3 3.50 Beta Adds Sybase](#)

[Simsoft Christmas Stocking](#)

[ClarionShop Big Prize Giveaway](#)

[New Product: cwAdvantage](#)

[xWordCOM Library 1.3](#)

[IP Driver and Data Server Now Shipping](#)

[xRuntimeStyle Manager v1.8](#)

[New DCT2SQL Template](#)

[Clarion Developers Challenge Week 9 Winner](#)

[Sterling Data Named Third-party Vendor of the Week](#)

[New Super Stuff and Super Invoice Released](#)

[vuFileTools Trend Micro Issue Resolved](#)

[ADDA 1.0.5](#)

[WinEvent 3.24](#)

[Berthume Software Named Third-party Vendor of the Week](#)

[BLAT 1.3](#)

[Icetips Xplore 1.205](#)

[XPTHEME 2 For Clarion 6.1](#)

[EasyMultiTag 2.07 For C61 9029](#)

[PowerOffice Home Page Moved](#)

[dpQuery 2.01](#)

[xDataBackup Manager Pro 2.0](#)

[xDataBackup Manager Lite 2.0](#)

[Advertising On Clarionfoundry](#)

[dpQuery 2.00](#)

[Ingress ODBC Patch](#)

[Riebens News Server](#)

[DOS Printer 9.05](#)

[DomainWatchDog - 100% Clarion Code](#)

[Capesoft In The UK](#)

[C6 Icon Extractor](#)

[Product Specific News Server](#)

[Clarion Utilities \\$49](#)

[Clarionfoundry RSS Feed](#)

[Postgres 8.0 Beta 3 Dev 1 Released](#)

[**Search the news archive**](#)

Legend

MS Subscription or free membership required

SO Subscription required

Clarion Magazine

DevCon 2004: Clarion ASP.NET

by James Cooke

Published 2004-11-05

One technology that many Clarionites are yearning for is a toolset that produces high-end web applications. There are currently several tools that provide web-based solutions, namely Internet Connect, Clarion ASP and ClarioNet. Each has strong points and each has weaknesses:

- Internet Connect provides virtually seamless HTML generation and allows native Clarion to be used in the generation of pages, but it lacks flexibility and the ability to configure the user interface using WYSIWYG.
- Clarion ASP produces superb HTML and JavaScript code, but the user interface does not have a WYSIWYG editor, the application's flow of logic is very tightly bound to the dictionary, and there isn't much that can be done in the way of embedded code.
- ClarioNet, in my opinion, provides the best functionality of all, with its native Windows user interface, but it requires a minimal proprietary download and it, like Internet Connect, requires Soft Velocity's proprietary Application Broker to be installed on the web server – which can be a difficult sell with larger commercial ISPs.

So why am I waffling about three other products instead of telling you about Clarion ASP.Net? Simple: In my opinion, this upcoming product promises to resolve all of the objections to the above three products.

Clarion ASP.NET will provide the following essential features of development:

1. A strong data layer with a familiar syntax.
2. Integration with the data dictionary, including relationships and validation rules.
3. A loosely coupled template set, based on HTML components (page,

- button, list box, radio, drop list etc)
4. Use of the window formatter (Ctrl+F) to design WYSIWYG web pages.
 5. Integration of existing technologies – in this case the .NET framework. The ASP.NET framework is a set of classes like the ABC classes, except they are 100 times bigger and richer. Also using the ASP.NET service, instead of Internet Connect, provides a much more reliable backend.
 6. Fully functional embed points for all server side events for all controls. This means, for example that you can trap a mouse click on the client and execute code for it. Overall, you have the same embed paradigm as the ABC and Clarion templates.
 7. Full use of the .NET runtime, which includes a magnificent and extended implementation of object orientation.

If SoftVelocity delivers even half of what was shown at DevCon 2004, Clarion ASP.NET will be a phenomenally successful product.

[James Cooke](#) works for Farm Credit Bank of Texas, and is currently developing several systems on an AS/400. He has been developing Clarion applications since 1992, and lives in Austin, Texas with his wife and two young children. For relaxation, he enjoys medium format photography, balanced with a liberal dose of sun next to the pool!

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 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.





Quick Links

[HOME](#)

[Subscribe/Renew](#)

[Clarion Books](#)

[FAQ \(Frequently Asked Questions\)](#)

[Log In](#)

[Download PDFs](#)

[Free Membership](#)

[Free Articles](#)

[Advertise](#)

[Reader Comments](#)

[Write For ClarionMag](#)

[Privacy](#)

[Contact](#)

Clarion Magazine

Using Clarion's Business Rules, Part 3

by Nardus Swanevelder

Published 2004-11-05

Last time I covered the two methods you need to be able to add your own rules at runtime. In this article I am going to look at what you need to do to implement the methods at runtime.

Queues

If you read my article [Translation, Clarion Style](#) you will see that I populate a global queue with the translation strings. To solve the problem of applying the rules at runtime I use the same technique. Create two global queues as follows:

```
RulesQueue          QUEUE , PRE( RQ )
Name               STRING( 255 )
Description        STRING( 255 )
Expression         STRING( 255 )
ControlUse         STRING( 255 )
Offset              BYTE
END
```

```
RulesControlQueue  QUEUE , PRE( RCQ )
Name               STRING( 255 )
ControlUse         STRING( 255 )
Action              BYTE
END
```

A word of caution: this technique can potentially cause a problem. If two threads are reading the queue at exactly the same time, but retrieving different values, then there may be data corruption. This is unlikely, since the

queues are being read from forms, but it could happen in theory. There are a number of solutions. One is to use the In-Memory-Database-Driver (IMDD) to replace the queue structure, since IMDD is thread safe. You could also use a class to copy the contents of the queues to thread-specific queues at the start of each thread, or you could also use two physical tables, and just clear them out at the start of each session. The problem with the last option is that you will generate unnecessary traffic if you are running your application over the network. To accommodate those who do not own IMDD I will continue using the queue technique.

```

Access:Rules.Open()
Access:Rules.UseFile()
Free(RulesQueue)
RUL:Name = ''
Set(RUL:Name_Key, RUL:Name_Key)
Loop
  If Access:Rules.Next() <> Level:Benign THEN BREAK.
  RQ:Name      = Clip(RUL:Name)
  RQ:Description = Clip(RUL:Description)
  RQ:Expression   = Clip(RUL:Expression)
  RQ:ControlUse  = Clip(RUL:Control)

```

If the user entered the control with a leading ? remove it.

```

If Sub(RQ:ControlUse,1,1) = '?'
  RQ:CONTROLUse = Sub(RQ:ControlUse,2,Len(Clip(RQ:ControlUse)))
End
RQ:Offset      = Clip(RUL:Offset)
Add(RulesQueue,RQ:ControlUse)
  If error() then stop(error()).
End
Access:Rules.Close()

Access:ControlsPerRule.Open()
Access:ControlsPerRule.UseFile()
Free(RulesControlQueue)
CON:Name = ''
Set(CON:Name_Key, CON:Name_Key)
Loop
  If Access:ControlsPerRule.Next() <> Level:Benign THEN BREAK.
  RCQ:Name      = Clip(CON:Name)
  RCQ:CONTROLUSE = Clip(CON:Control)

```

If the user entered the control with a leading ? remove it.

```

If Sub(RCQ:ControlUse,1,1) = '?'
    RCQ:CONTROLUse = Sub(RCQ:ControlUse,2,Len(Clip(RCQ:ControlUse)))
End
RCQ:RuleAction = Clip(CON:RuleAction)
Add(RulesControlQueue,RCQ:Name)
If error() then stop(error()).
End
Access:ControlsPerRule.Close()

```

Now you have the rules available in a global queue. The next question is how do you populate these on a procedure? I have tried various things but in the end settled on the following: I create a local queue and populate it with all the controls on the window, then I go through the rules queue and check to see if the control in the rule is used on the window. If it is used I add the rule, and then add the controls linked to that rule.

How do you populate the local queue? The easiest way is to use a template. The code is as follows:

```

Free(LocalWindowQueue)
#FOR(%Control)

```

Some controls do not have a %ControlUse variable, so I populate the ControlUse with the control after I remove the ?. For example the ?OK control does not have a ControlUse variable so I remove the ? and ControlUse becomes OK.

```

#IF(%ControlUse = '')
LWQ:ControlUse = Upper(Sub('%control',2,Len('%control')))
#Else
LWQ:ControlUse = Upper('%ControlUse')
#endif
LWQ:Control = %Control
Add(LocalWindowQueue,LWQ:ControlUse)
#endiffor

```

You could also use Prop:NextField and Prop:Use to populate the queue.

See my article [Adding Page of Pages to a Clarion Report](#) for an example of using Prop:NextField.

You now have a queue with all the window controls and ControlUse variables.

Next you have to run through the rules to determine if the rule's control is used on this window.

```

!Run through rules
Loop I# = 1 to Records(RulesQueue)
    Get(RulesQueue,I#)
    If not error()
        !Check if rule's control is on screen
        LWQ:ControlUse = RQ:ControlUse
        Get(LocalWindowQueue,LWQ:ControlUse)
        If not error()
            !If rule's control is on screen - add rule
%RuleBaseName.AddRule(Clip(RQ:Name),Clip(RQ:Description),|
    Clip(RQ:Expression),LWQ:Control,RQ:Offset)
            !Check rule's rule-action-control
            RCQ:Name = RQ:Name
            Get(RulesControlQueue,RCQ:Name)
            If not error()
                Position# = Pointer(RulesControlQueue) + 1
                Loop
                    !If rule's rule-action-control is on window -
                    ! add control to rule
                    LWQ:ControlUse = RCQ:ControlUse
                    Get(LocalWindowQueue,LWQ:ControlUse)
                    If not error()
%RuleBaseName.AddControlToRule(Clip(RCQ:Name),LWQ:Control,|
                        RCQ:RuleAction)
                    End
                    Get(RulesControlQueue,Position#)
                    If error() or RCQ:Name <> RQ:Name then break.
                    Position# += 1
                End
            End
        End
    End
End

```

That's it, you have added the rules to the procedure. I have created a

template that will do most of this for you automatically. You can obviously move all of the code to the template yourself.

But as they say on television, that is not all, if you phone now... Just joking - the one thing that is still outstanding is the question of how to override/disable a rule on a specific procedure.

Just add another table, another global queue, another procedure and some code and you are done.

```
OverrideProcPerRule
RuleName      String(255)
RuleProcedureName   String(255)

RulesProcQueue          QUEUE , PRE(RPQ)
Name        STRING(255)
ProcName     STRING(255)
                           END
```

And add the following code to the template:

```
RPQ:Name      = Clip(RQ:Name)
RPQ:ProcName = '%Procedure'
Get(RulesProcQueue, RPQ:Name, RPQ:ProcName)
If error()
!Add the rule
```

This will only give you the ability to disable a rule on a procedure but not to change it. I believe however that this will give you enough functionality to start play with dynamic business rules.

Implement the template

Copy the template to the template folder, it can be the Clarion folder or the 3rdparty\Template folder. Register the template in Clarion. You do not have to load the Clarion Business Rule Manager templates. My template adds all the necessary code.

Add the global extension

1. Open your app and go to the global settings.
2. Click on the Extensions button and then click on the Insert button.
3. Scroll down until you get to Rules Template and then click on Global External Business Rules Manager.
4. Complete the Global Rule Description
5. You can leave all the other prompts as the default values are sufficient for now.

Add the local extension

1. Open the procedure where you want to be able to process rules at runtime. Click on Extensions, choose External Rules Local.
2. You can leave all the prompts as the default values are sufficient for now.

Run the application

1. Start the application, click on Browse, and then choose Rules.
2. Add a rule. For example::
 - **Name:** Gender
 - **Description:** Gender must be M or F
 - **Expression:** PEO:Gender = 'F' or PEO:Gender = 'M'
 - **Control:** PEO:Gender
 - **Offset:** 3
3. Add a control to the rule, click on Controls tab and insert a record.
 - **Control:** ?Ok
 - **Rules Action:** RuleAction:Disable
4. Add a procedure name where you do not want the rule to be enforced, click on the Procedure tab and insert a record
 - **Procedure name:** Update_rules

5. Close the form and on the browse click on the refresh button. This will refresh the global queues with the new rule. Close the browse and test the new rule on the Browse the People file update form.

Disadvantages

This approach has three disadvantages that I know of:

- It always instantiates RulesCollection class and RulesManager class even if there are no rules applicable to the procedure
- It always instantiates a local queue and populates it with controls on the window, even if no rules are applicable on the window
- If SoftVelocity changes the Rule Manager you might have to change some of the functionality described in the series of articles. There is a rumor that the new Rules Manager provides four levels of severity rather than the original "broken/non-broken" pair. Once this is released, rule expressions based on "zero/non-zero" only may give unpredictable results.

Advantages

This approach also has several advantages:

- You could also use this as part of your Translation solution. Now all your business rules messages can be translated.
- You can change your rules at runtime – need I say more?

Summary

In this series I have shown you how you can extend Clarion's business rules functionality to use rules defined in tables. In this way you can supply new business rules (global or local to a procedure) to your customers without having to update the application itself – instead, you simply send along an updated set of rules tables.

[Download the source](#)

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

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 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.



Quick Links

[HOME](#)

[Subscribe/Renew](#)

[Clarion Books](#)

[FAQ \(Frequently Asked Questions\)](#)

[Log In](#)

[Download PDFs](#)

[Free Membership](#)

[Free Articles](#)

[Advertise](#)

[Reader Comments](#)

[Write For ClarionMag](#)

[Privacy](#)

[Contact](#)

Clarion Magazine

A Configurable Expression Editor, Part 1

by Tim Phillips

Published 2004-11-05

Clarion does not have a good integral tool for defining an expression at runtime. This is inconvenient since expressions can be used to control filtering, sorting and variable assignments. Various third party solution exist, such as [NiceTouch QueryWizard](#) and [Fomin Tools](#) InvokeDictionary control template, but they do impose a specific look and feel and often require adding a DLL to the application to support the expression builder.

When attempting to upgrade a utility that I use, I ran into these issues once again. This time I was able to combine some techniques lifted from my [Basic Editor](#) with a local procedure, some queues and a couple of helper procedures to create "the perfect fix" for my problem.

Problem Definition

What were my needs in an expression editor? In order of importance they probably were:

- 1) Room to see the entire expression I am editing. I tend to use verbose variable names and a complete expression is hard to comprehend if I have to scroll up and down a lot to view it (this is one of my pet peeves about the Clarion ReportWriter expression editor).
- 2) Easy lookup/insertion of the various components that can be used in a given expression. I am forever forgetting if the "string to be looked for" is the first or second argument when using Instring(). I want to see an "good description" of what a component like Instring() is and be easily able to add it to my expression where I need it.

- 3) Compactness. I don't want to have to alter an application a lot to add an expression editor. I'd prefer to not have to add a separate DLL, and I don't want something that is hard to install or set up.
- 4) My target audience is "power users". These are people who are not intimidated by writing an expression by hand. They do not need wizards or an interface that lays all the options onto a single window so they can see them. They probably already know most of the Clarion functions and operators and just need a mental jog for one they have forgotten or to glance at the list of available data fields to remember if the field is LastName **or** OwnersLastName.

User Interface

Needs 1 and 2 and 4 made it pretty clear that my biggest issues with existing expression editors were concerned with the user interface. This was one of those occasions when I go low-tech, grabbing a stack of scrap paper and a pen and just doodling different interface concepts in a cryptic scrawl so abstract it probably qualifies as modern art. Good ideas get pushed to one side to be incorporated into a following sketch; the bad ideas get tossed back into the recycle bin, literally.

After a while, what emerges from this exercise is definitely "modernistic" and minimal.

My final interface is just a window with a big text editing field and four buttons. Figure 1 shows the "final product" as built.



Figure 1. The Expression Editor

Functional Concept

Part of the interface is the concept of how it will work. Some of that is visible from the image above. The large text field allows me to edit a rather lengthy expression easily.

I know from my work on the Basic Editor that I can alert keys and use those to call three distinct lookups (Data Fields, Functions, Operators) to insert predefined values into the text field when I need them. This lets me keep my hands on the keyboard so I can work quickly (and allow me to maximize the space on the window devoted to the text field for editing).

If I pass the expression to be edited into the editor procedure and assign it to a local string to be displayed, I can control what is returned from the procedure. I can use the CLOSE control template to return the edited value, and the CANCEL control template to return the original value passed into the procedure.

I can use the Evaluate() function from Clarion to check the validity of the expression (as long as the variables used are Binded before the Expression Editor is called), so I can tell in a moment if I have a syntax problem.

And it is easy to set the local expression string to a blank with the click of a button.

That's the general idea of the Expression Editor. Next week I'll explain the implementation.

[Tim Phillips](#) began programming Clarion with Version 3.0 for DOS. He currently works with Clarion 5.5/6 ABC. His preferred programming technique involves a lot of bass rock guitar on his cordless headset and vigorous application of the Keep It Simple Stupid principle. When not programming, he can be found trying to write fiction, "building Legos" with his two nieces, or being obsessive about television shows that have gone off the air.

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 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.





Quick Links

[HOME](#)

[Subscribe/Renew](#)

[Clarion Books](#)

[FAQ \(Frequently Asked Questions\)](#)

[Log In](#)

[Download PDFs](#)

[Free Membership](#)

[Free Articles](#)

[Advertise](#)

[Reader Comments](#)

[Write For ClarionMag](#)

[Privacy](#)

[Contact](#)

Clarion Magazine

Clarion Magazine's Podcasts

What's a Planet Clarion Podcast, you ask? For starters, try this podcast [definition](#) from Wikipedia. Basically we're talking about audio programming in MP3 format, which means that all you really need to do is click on one of the links below to download and listen to Planet Clarion. The term *podcasting* actually refers to the combination of audio files, an iPod, and software that lets you automatically download these MP3s to your iPod.

Again, you don't *need* any fancy software to listen to these MP3s. But if you want to automatically download the feed instead of coming to this web page and clicking on the links, try some of the RSS software listed at [iPodder.org](#). Point the RSS reader of your choice at our Planet Clarion RSS feed:

<http://www.clarionmag.com/planetclarion.rss>

Comments? Send us an [email](#)

Planet Clarion is hosted by Dave Harms, Clarion Magazine's editor, and Andrew Guidroz II, your favorite Cajun. We'll have more information about Planet Clarion on this page as the Planet matures. Meanwhile, enjoy this listing of recent Planet Clarion, um, pseudo-podcasts.

Please note that each podcast, in its entirety, is always freely available. Some files may be quite large. If you don't want to download the entire podcast, you can also download individual tracks (these, however, are not available via RSS, at least not yet). Two conditions apply: first, you must be have a subscription that covers the track you want to download, and second, these are simple extracts from the original podcast and may start and end abruptly, and occasionally insensibly.

You may also want to download the entire podcast, and use the track list to

quickly locate a particular subject of interest.

Planet Clarion for November 26, 2004

Track	Start	Length	Size	Description
<u>Complete Podcast</u> (Free Access)	00:00:00	40:42:00	19328 K	Planet Clarion #2 is in the can! In this edition Andrew and Dave look at color theory and application skinning, discuss the IP driver, and talk to graphic designer, Clarion developer, and rising photography star Leroy Schulz.

Individual tracks

<u>Track 1</u> SO	00:00:00	00:05:08	2408 K	Intro, and a bit of information about how we record the podcast.
<u>Track 2</u> SO	00:05:08	00:02:53	1360 K	Recording Skype, and Andrew's live lunar eclipse report
<u>Track 3</u> SO	00:08:01	03:08:00	1472 K	Is that a Clarion application, or a ransom note?
<u>Track 4</u> SO	00:11:10	00:05:25	2542 K	There's more than one way to skin an application
<u>Track 5</u> SO	00:16:35	00:06:07	2872 K	Color theory and making your app look cool
<u>Track 6</u> SO	00:22:42	00:04:53	2294 K	Leroy Schulz, graphic designer to the stars (well, the planets, anyway)
<u>Track 7</u> SO	00:26:36	00:03:17	1544 K	Leroy's second funkiest domain name
<u>Track 8</u> SO	00:30:53	00:07:19	3438 K	The IP driver, and what ever happened to Bob Foreman's tie?

Track 9 	00:38:13	00:01:24	660 K	Keep those cards and letters coming
Track 10	00:39:38	00:01:04	503 K	Commercial: Never Better!

Planet Clarion for November 9, 2004

Track	Start	Length	Size	Description
Complete Podcast (Free Access)	00:00:00	00:34:00	16813 K	In this first ever Planet Clarion podcast, hosts Dave Harms and Andrew Guidroz II discuss topics ranging from "Why stay with Clarion?" to how many developers are using Clarion, and the impact of Clarion.NET.

Individual tracks

Track 1 	00:00:00	00:01:07	522 K	Intro, with some theme music and a bit of rambling from Andrew and Dave
Track 2 	00:01:07	00:04:40	2185 K	Why stay with Clarion? Pre-release anxiety and the wonders of wizarded apps.
Track 3 	00:05:45	00:01:22	647 K	Making money by mopping up after other non-Clarion developers
Track 4 	00:07:09	00:03:47	1777 K	How many Clarion developers are on the Internet? Long live the lurker.
Track 5 	00:10:56	00:02:26	1143 K	DevCon 2004 and Clarion.NET
Track 6 	00:12:32	00:01:15	584 K	Rubber boots

Track 7 <small>SO</small>	00:14:37	00:11:14	5269 K	More .NET - it's gonna fark! .NET vs the Win API, exceptions, managed code...
Track 8 <small>SO</small>	00:25:50	00:07:34	3548 K	Clarion does it today. Grok it.
Track 9 <small>SO</small>	00:32:24	00:01:23	650 K	Closing comments
Track 10	00:34:48	00:01:04	502 K	Commercial: "Never Better!" (free access)

Freedom to distribute podcasts

You are free to distribute *public access* podcasts from Clarion Magazine, provided you do not modify those podcasts, and you do not charge any fees for the podcasts. In other words, if you want to put a podcast up on your server, feel free.

You may not, however, distribute individual tracks without express permission from Clarion Magazine.

Copyright © 1999-2004 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.



Quick Links

[HOME](#)

[Subscribe/Renew](#)

[Clarion Books](#)

[FAQ \(Frequently Asked Questions\)](#)

[Log In](#)

[Download PDFs](#)

[Free Membership](#)

[Free Articles](#)

[Advertise](#)

[Reader Comments](#)

[Write For ClarionMag](#)

[Privacy](#)

[Contact](#)

Clarion Magazine

A Configurable Expression Editor, Part 2

by Tim Phillips

Published 2004-11-19

In [Part 1](#) I defined the user interface and functional concept of my expression editor. Now it's time to get down to the actual code, beginning with a queue declaration.

Defining a Global Queue

After playing with various "cute" ways to pass a list of data fields and functions that are allowable within a specific expression into the `ExpressionEditor` procedure, I went "simple" and decided to rely upon a global queue. Accordingly, I defined a global "table" in the Data Dictionary called `GlobalQueues` with a prefix of `GLQ`. Within this, I declared a queue called `ExpressionEditorQ` with a prefix of `EEQ` and a Storage Class of Thread. I then defined four fields within the queue:

- Value as a string of 80
- Description as a string of 80
- Type as a string of 40
- Note as a string of 1000

Defining the `ExpressionEditor` Procedure

I fabricated the `ExpressionEditor` procedure itself by using the Window - Generic Window Handler template from the Class ABC templates. I defined a local variable called `LExpression` as a string of 5000 characters to hold the

expression to be edited. The prototype for the ExpressionEditor procedure needs to be set to (STRING IExpression), STRING with a Parameters value of (STRING IExpression) and Return Value set to LExpression. This permits passing a string representing an expression into the procedure and returning it when completed.

The LExpression variable needs to be populated as a text field onto the ExpressionEditor window and have the vertical scroll bar activated. The F4 key, right mouse key, Ctrl-F, Ctrl-D and Ctrl-O key combinations all need to be alerted for the LExpression field to drive the rest of the expression editor. The Close and Cancel control template buttons, and two "regular" buttons labeled Check If Valid and Clear, need to be added to this window to complete its basic construction.

A little polishing (establishing a good window size, adding tooltips, putting some prompts on the window to "jog the memory" of what keystrokes does what) and it is time to install the "real code" to make things happen.

1) Place the following code at the Local Data Other Declarations embed:

```
LC CLASS ! declare local class
  CallBrowseExpressionEditorQ Procedure (string LimitDisplayTo)
END
CR Equate('<13><10>') ! declare equate for carriage return
```

The first part defines a local class I can use to control displaying the lookup windows for Data Fields, Functions and Operators. The second part gives me a constant I can use in place of typing <13><10> when trying to define a carriage return when creating a string to serve as the Note for a record in the GLQ:ExpressionEditorQ queue.

2) Embed the following code at the Control Events => ?Cancel => Accepted => embed, above the Generate Code point:

```
LExpression=IExpression
```

This resets the local value of the expression to the value that was passed in. Since LExpression is the returned value from the procedure, pressing Cancel effectively erases any change made to the expression within the ExpressionEditor procedure.

3) At Control Events => ?Clear => Accepted => above the Generate Code embed add this code:

```
LExpression= ''Display( ?LExpression)
```

This resets the local value of the expression to nothing and refreshes the window display so the user has a clean slate to work upon.

4) At Control Events => ?LExpression => AlertKey => above the Generated Code embed add this code:

```
case KeyCode() ! what key has been pressed
of F4Key orof MouseRight ! F4 or right-mouse
    ! display popup menu
    Execute Popup('Select DataField Ctrl-D|' |
                  & 'Select Function Ctrl-F|Select Operator Ctrl-O')
    LC.CallBrowseExpressionEditorQ('Fields')
    LC.CallBrowseExpressionEditorQ('Functions')
    LC.CallBrowseExpressionEditorQ('Operators')
END
of CtrlD ! ctrl+D
    LC.CallBrowseExpressionEditorQ('Fields')
of CtrlF ! ctrl+F
    LC.CallBrowseExpressionEditorQ('Functions')
of CtrlO ! ctrl+O
    LC.CallBrowseExpressionEditorQ('Operators')
end
```

This code powers the alert keys set on the LExpression field. The Case KeyCode() will wait for alerted keystrokes, and cause the correct actions to be taken depending upon which combinations have been pressed. I have provided for displaying a popup menu of options when F4 is pressed, or directly calling the various lookups with Ctrl-D, Ctrl-F and Ctrl-O.

The LC.CallBrowseExpressionEditorQ() method is used to call for the display of the correct type of lookup list based upon the parameter passed into it.

5) At Control Events => ?Valid => Accepted => above the Generate Code embed, add this code:

```
if CLIP(LExpression)>''  
  if Evaluate(LExpression)= ''  
    Message('Expression Is Not Valid')  
  else  
    Message('Expression Is Valid')  
  end  
end
```

The Clarion EVALUATE() function processes an expression and returns a string holding the value of the expression. If an expression can not be processed - the syntax is incorrect - then it returns an empty string'. So long as all the fields and programmer-defined functions used in an expression are bound before the ExpressionEditor procedure is called, the EVALUATE() function as used above will indicate immediately if the syntax of the statement is correct.

6) At Local Objects => ABC Objects => Window Manager => Init => Code below Initialize the Procedure add this code:

```
! assign passed in parameter to the local copy  
LExpression=IExpression  
! load various operators  
do AddToExpressionEditorQForOperators  
! load various functions  
do AddToExpressionEditorQForFunctions
```

This initializes the local version of the expression string with the value passed to the procedure as a parameter, and calls two routines that load the GLQ:ExpressionEditorQ queue with operators and functions that I consider "standard" and want any instance of the ExpressionEditor procedure to have access to.

7) At Procedure Routines add this code:

```
AddToExpressionEditorQForOperators Routine
ATEEQ('Operators','&','concatenate two strings together','')
ATEEQ('Operators','[position1,position2]',|
      'slice from position 1 to position 2 in a string','')
ATEEQ('Operators','+','add two numbers','')
ATEEQ('Operators','-','subtract two numbers','')
ATEEQ('Operators','*','multiply two numbers','')
ATEEQ('Operators','\','divide two numbers','')
ATEEQ('Operators','=','two values are equal','')
ATEEQ('Operators','<','value on the left is less than the ' |
      '& value on the right','')
ATEEQ('Operators','>','value on the left is greater than the ' |
      '& value on the right','')
ATEEQ('Operators','<>','value on the left is not equal to ' |
      '& the value on the right','')
ATEEQ('Operators','>=','value on the left is greater than or ' |
      '& equal to the value on the right','')
ATEEQ('Operators','<=','value on the left is less than or ' |
      '& equal to the value on the right','')
ATEEQ('Operators','AND','do a boolean AND of the values to ' |
      '& the left and right','')
ATEEQ('Operators','OR','do a boolean OR of the values to the ' |
      '& left and right','')
ATEEQ('Operators','(',')','left parenthesis','')
ATEEQ('Operators','')','right parenthesis','')
ATEEQ('Operators','()'','left and right parenthesis as a set','')
ATEEQ('Operators','Choose(expression,if true,if false)',|
      'Choose one option or another','')
ATEEQ('Operators','Choose(number,value1,value2...valueN)',|
      'Choose option based upon value of number','')
AddToExpressionEditorQForFunctions Routine
ATEEQ('Functions','Today()', 'return number that is # of days ' |
      '& since January 1, 1801 and today','')
ATEEQ('Functions','CLIP()','clip blank spaces off end of a string','')
ATEEQ('Functions','Left()','left justify the text in a string','')
ATEEQ('Functions','Right()','right justify the text in a string','')
ATEEQ('Functions','Center()','center justify the text in a string','')
ATEEQ('Functions','Chr()', 'return the ASCII character for a ' |
      '& specific ASCII code',('Example:' & CR & |
      'CHR(13) will cause insertion of a carriage return into a string'))
ATEEQ('Functions','Clock()', 'return time now as hundredths ' |
      '& of a second since midnight, plus one','')
ATEEQ('Functions','Date(month,day,year)', 'return date from month, ' |
      '& day year','')
```

```

ATEEQ( 'Functions' , 'Day(date)' , 'return day portion of a date' , '' )
ATEEQ( 'Functions' , 'Month(date)' , 'return month portion of a date' , '' )
ATEEQ( 'Functions' , 'Year(date)' , 'return year portion of a date' , '' )
ATEEQ( 'Functions' , 'Upper(string)' , 'string converted completely ' |
& ' to uppercase' , '' )
ATEEQ( 'Functions' , 'Lower(string)' , 'string converted completely ' |
& ' to Lowercase' , '' )

```

These two routines will use the ATEEQ procedure (which I will explain shortly) to add records to the GLQ:ExpressionEditorQ queue. Each record represents an operator or a function that I wish to be able to select from a lookup when running the ExpressionEditor procedure. Notice that I have deliberately culled the list of common Clarion functions to only have those functions that I use a lot. The remaining functions still work if I type them, they are just not something I want at my finger-tips in the lookups.

8) At Local Procedures add this code:

```

LC.CallBrowseExpressionEditorQ Procedure (string LimitDisplayTo)
ReturnValue String(80)
Code
ReturnValue=SelectFromExpressionEditorQ(LimitDisplayTo)
if CLIP(ReturnValue)<>'xCANCELx'
  SetClipboard(CLIP(ReturnValue))
  PressKey(CtrlV)
end

```

This is the definition of the single method I declared in my local class at 1) above.

This particular method is passed a string defining what type of values (Operators, Functions, DataFields) the user wants to select from. That is passed as a parameter to the procedure SelectFromExpressionEditorQ (see below). SelectFromExpressionEditorQ will either return a selected value or it will return the string XCANCELX - signifying that user chose to cancel the selection. If a selected value is returned, the method will load the value into the windows clipboard and then use PressKEY() to insert the value from the clipboard into the text field at the current location of the cursor.

NOTE: Local classes are something that I have only started to use in the last 10-12 months and instances like this show the power they can unleash. Notice how "clean" this code is when used in the ExpressionEditor procedure. You could use a routine in place of this class, but you would have to define another local string within the procedure to handle LimitDisplayTo and you would have to initialize it before calling the routine. By using a class, you end up with a single line of code instead of two to start each lookup. Start thinking of local class methods as routines on steroids and you will start using them everywhere.

Defining the ATEEQ procedure

The ATEEQ procedure is a source procedure with a prototype of (STRING IType, STRING IValue, STRING IDescription, STRING INote) and a Parameters of (STRING IType, STRING IValue, STRING IDescription, STRING INote). Be sure to check the Declare Globally checkbox. This will let you use it all through the program; if you don't do this you'll have to add the procedure to the Procedures list for each procedure you want to use it in (think of it as declaring the ATEEQ procedure as a built-in Clarion function, one that you have written and control the behavior of). In the Processed Code embed for the procedure add this code:

```
EEQ:Value=IValue
EEQ:Description=IDescription
EEQ>Type=IType
EEQ>Note=INote
add(GLQ:ExpressionEditorQ)
```

This code is very simple. It just equates input parameters to the various fields in the GLQ:ExpressionEditorQ and then adds another record to that queue. By encapsulating this action within a global procedure, I make it into something I can reuse over and over in other procedures across the application. This will become important once I have finished building the

ExpressionEditor Procedure and move on to actually using it in something like a Report Procedure to build the filter statement for the Report.

Defining the SelectFromExpressionEditorQ procedure

SelectFromExpressionEditorQ is the procedure that will serve as the lookup for Data Fields, Operators or Functions. It is created from a Window template with a prototype of (STRING TypeToLimit), STRING and a Parameters of (STRING TypeToLimit).

Some local variables need to be defined within the procedure itself:

```

ReturnString      STRING( 80 )
I                LONG
LocalQ           QUEUE , PRE( LQ )
Value            STRING( 80 )
Description      STRING( 80 )
Note             STRING( 1000 )
END

```

ReturnString should be set as the Return Value for the Procedure

The window should be laid out to look like Figure 2.

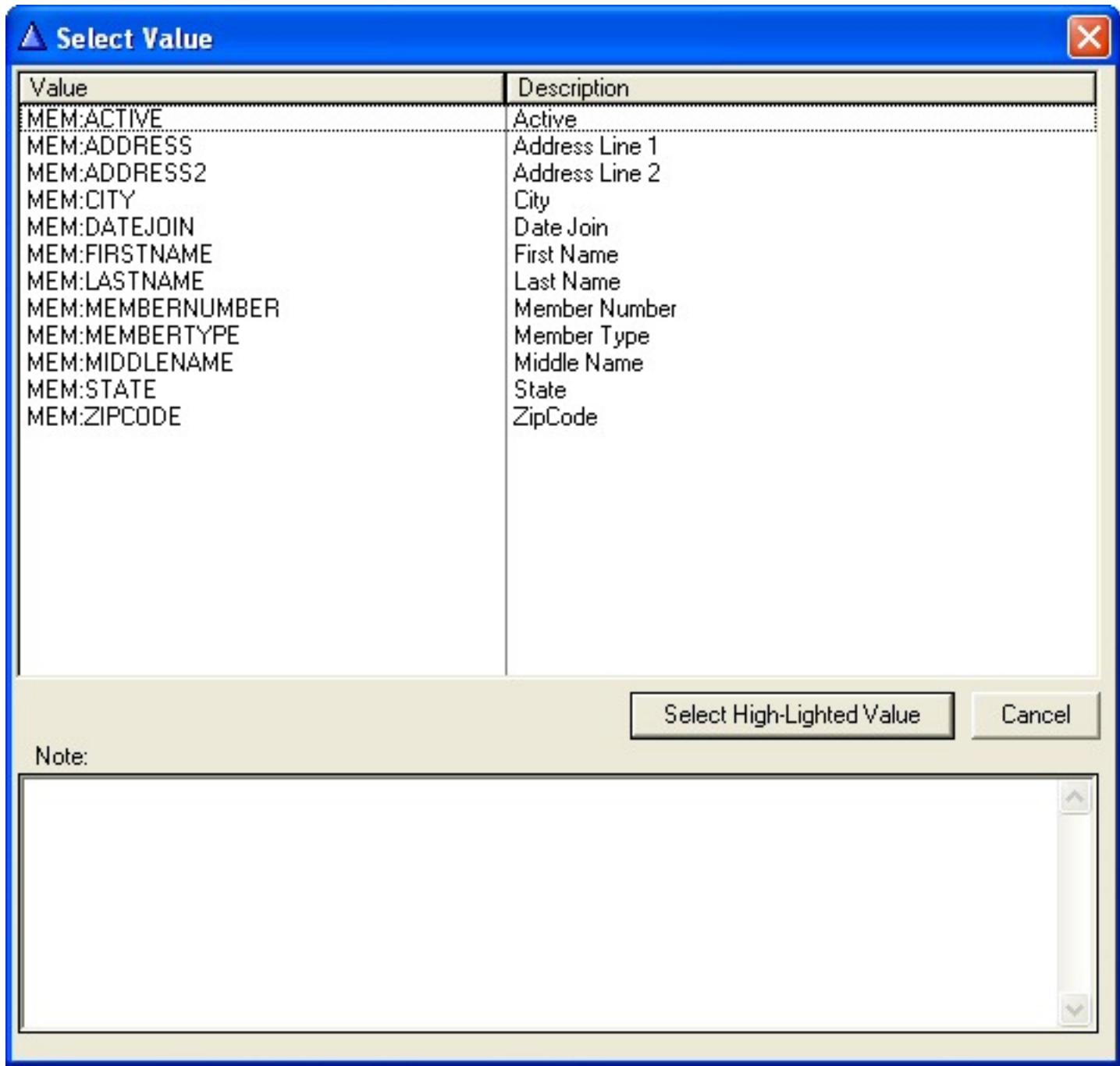


Figure 2.The SelectFromExpressionEditorQ window

The listbox has the From defined as LocalQ, Use variable defined as ?List1, MouseLeft2 alerted, and is formatted to display the fields Value and Description from LocalQ in the local data.

Activate a Timer for this window by going to Window Properties on the Extra tab and typing 100 into the Timer entry. This will be used to "refresh" the value of the Note displayed on the window as the window is scrolled by the user.

Five chunks of embedded code are needed to make this procedure run so it can display the correct lookups and permit them to be selected.

1) at Window Events => Timer above the Generated Code add this:

```
Get(LocalQ,Choice(?List1))
Display(?LQ:Note)
```

On each occasion that the window Timer fires, this will cause the currently high-lighted LocalQ record to be retrieved, and the Note field on the window to be refreshed for display so you can see its contents. This effectively makes the Note field "hot" even though it is in a queue. With a short enough Timer value, there is no noticeable pause between selecting a record and seeing the Note fill in. I have definitely seen worse with a memo in a Topspeed file on a busy network.

2) At Control Events => ?Cancel => Accepted above Generated code add this:

```
ReturnString='xCANCELx'
```

This will set ReturnString to the predefined value indicating that the user has canceled selecting a value from the lookup procedure, just before the built-in code of the Cancel button template fires to close the window down and return the user to the ExpressionEditor procedure.

3) At Control Events => ?List1 => Alert Key below the Generated Code add this:

```
Case KeyCode()
of MouseLeft2
  Get(LocalQ,Choice(?List1))
  ReturnString=LQ:Value
  Post(Event:CloseWindow)
end
```

This code will cause the lookup procedure to return the Value field for the highlighted LocalQ record if the user double-clicks on the listbox.

4) At Control Events => ?SelectHighLightedValue => Accepted above the Generated code add this:

```
Get(LocalQ,Choice(?List1))
ReturnString=LQ:Value
Post(Event:CloseWindow)
```

This code will cause the lookup procedure to return the Value field for the high-lighted LocalQ record if the user presses the Select Highlighted Value button on the window

5) At Local Objects => ABC Objects => Windows Manager => Init => Code => below Initialize the Procedure add this:

```
Loop I=1 to Records(GLQ:ExpressionEditorQ)
  Get(GLQ:ExpressionEditorQ,I) ! get each record
  if CLIP(EEQ:Type)=CLIP(TypeToLimitTo)
    ! add global values to the local Q to populate it
    LQ:Value=EEQ:Value
    LQ:Description=EEQ:Description
    LQ:Note=EEQ:Note
    Add(LocalQ)
  end
end
Sort(LocalQ,+LQ:Value) ! sort local queue properly
```

This code loops through the global queue filled with definitions of Data Fields, Functions and Operators. If Type (Fields,Functions,Operators) matches what the user wants to select from on this occasion of calling the SelectFromExpressionEditorQ procedure, the global queue record is copied into LocalQ. LocalQ is then displayed for the user to make a selection.

Using the ExpressionEditor Procedure

The ExpressionEditor and its two attendant procedures should now

compile. You are ready to actually use the ExpressionEditor.

To build an example, use the members.TPS file from the EventMgr example program that ships with Clarion. Import it into your dictionary, and use the wizards to create a browse and form to maintain the members.TPS file and a report to print it out. The PrintMembers report is what you will "hot-rod" so you can edit the filter it uses at runtime.

In Local IData you need to create a variable called LFilterExpression as a string of 5000. This will hold the filter expression you are going to create with the ExpressionEditor.

NOTE: It is important that the length of LFilterExpression here match the length of LExpression in the ExpressionEditor procedure. If they do not match, it is possible that a filter expression will get truncated as it is moved into and out of the LExpression variable in the ExpressionEditor.

At Local objects => ABC Objects => Window Manager => Init => Code below the Prepare Alert Keys add this:

```

Free(GLQ:ExpressionEditorQ) ! free the global queue to empty it out
! add all the fields to the global queue
ATEEQ('Fields','MEM:MEMBERNUMBER','Member Number','','')
ATEEQ('Fields','MEM:LASTNAME','Last Name','','')
ATEEQ('Fields','MEM:FIRSTNAME','First Name','','')
ATEEQ('Fields','MEM:MIDDLENAME','Middle Name','','')
ATEEQ('Fields','MEM:ADDRESS','Address Line 1','','')
ATEEQ('Fields','MEM:ADDRESS2','Address Line 2','','')
ATEEQ('Fields','MEM:CITY','City','','')
ATEEQ('Fields','MEM:STATE','State','','')
ATEEQ('Fields','MEM:ZIPCODE','ZipCode','','')
ATEEQ('Fields','MEM:DATEJOIN','Date Join','','')
ATEEQ('Fields','MEM:ACTIVE','Active','1 if active. 0 if not')
ATEEQ('Fields','MEM:MEMBERTYPE','Member Type','','')
LFilterExpression='' ! default is no filter
! call editor
LFilterExpression=ExpressionEditor(LFilterExpression)
! apply filter to report via SetFilter method

```

```
ThisReport.SetFilter(CLIP(LFilterExpression))
```

This code will populate the GLQ:ExpressionEditorQ with all the fields the user should have access to when writing a filter. It will also attach a blank filter expression to the report procedure using the SetFilter() method.

By using this particular embed you can be certain that all the variables available have already been "bound" by the PrintMember procedure itself (so the Check if Valid button in the ExpressionEditor will work), and that the filter method has already been set by the PrintMember procedure so you can override it and know that the reassignment will work.

That is it.

Compile the program and you should be able to enter a filtering expression at runtime to limit the PrintMembers report to only those records you want.

Final Thoughts

This ExpressionEditor is quite simple to build (once you know the tricks), but is not really all that trivial a tool. It is not as polished as some of the third party tools I use, but it supplies a convenient and compact mechanism for defining expressions without being encumbering to a true power user.

I generally install the ExpressionEditor "deeper" than I have in the PrintMembers procedure. Usually, I have a browse/form combination which is called from the report and allows the user to select a complete package of filtering statements, sorting order statements, and potentially field assignments for usage with the report. The various fields on the update form then have access to the ExpressionEditor to help in defining them. This is part of a concept that I call Configurable Reports, and it is my intention to write that up as an article for the Clarion Magazine in the near future.

One of the really nice things about this ExpressionEditor is it can be

customized for your circumstances. The windows can all be altered to match the "look and feel" of the rest of your interface. Since the list of fields, functions and operators in the global queue is done with code, you can easily omit or add items as you wish. Indeed, since you have all the code here, you can modify it any way that you wish.

[Download the source](#)

[Tim Phillips](#) began programming Clarion with Version 3.0 for DOS. He currently works with Clarion 5.5/6 ABC. His preferred programming technique involves a lot of bass rock guitar on his cordless headset and vigorous application of the Keep It Simple Stupid principle. When not programming, he can be found trying to write fiction, "building Legos" with his two nieces, or being obsessive about television shows that have gone off the air.

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 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.



Quick Links

[HOME](#)

[Subscribe/Renew](#)

[Clarion Books](#)

[FAQ \(Frequently Asked Questions\)](#)

[Log In](#)

[Download PDFs](#)

[Free Membership](#)

[Free Articles](#)

[Advertise](#)

[Reader Comments](#)

[Write For ClarionMag](#)

[Privacy](#)

[Contact](#)

Clarion Magazine

Managing Report Page Breaks With The C6 Break Manager

by Geoff Bomford

Published 2004-11-26

In the [first of these articles](#) I looked at the concept of report breaks, and demonstrated how three of the most frequently asked questions about reporting in Clarion can be handled by the reporting techniques found in all versions of Clarion:

1. Where can I put the code that executes before the Break Header/Footer?
2. How can I perform calculations on totals, before they print in the Break Footer?
3. How can I re-print a Break Header at the top of a new page?

This article will demonstrate the powers of the new `BreakManagerClass` and show how these three tasks can be handled with Clarion 6.1.

The `BreakManagerClass` (BMC)

The management of breaks found in previous versions of Clarion, and discussed in my previous article, is now officially referred to as the "Traditional" reporting technique. The `BreakManagerClass` (hereafter referred to as BMC) is a completely new class and wrapper template, written by Softvelocity's Diego Borojovich, and first implemented in Clarion 6.0. It was improved in Clarion 6.1 and is likely to evolve with even more power in future releases of Clarion. At this stage it is only available for the ABC template chain, and since it is derived from the `ReportManagerClass` it might not be implemented in the Clarion template chain.

The online help gives a succinct overview of this class, which is worth repeating here.

The BreakManagerClass handles embedded break events for a target report. Each break can perform totaling based on a data element's contents changing. Breaks can be nested, allowing the contents of one break result to determine another break result. Conditional headers and footers can be printed by any break. Each break is totally customizable through available embed points defined in virtual methods.

In traditional reporting, you manage breaks by means of nested detail bands; when a specified value in a parent detail changes, a new set of child details is printed. The actual detail printing is managed within the black box of the report engine, which is why it's often so difficult to get reports to print the way you want. The BMC takes over that task from the print engine, and gives you greater control over the printing process.

At the time of writing this article, the online help for this class is somewhat sparse, so I'll be taking a detailed step-by-step look at how this class can be used.

Formatting Break Headers

Formatting break headers should be the easiest task to perform, but before I begin, I want you to discard *all* your previous notions of how Clarion reporting works! The BMC is a completely different concept.

C61Report1 procedure

BMC reports are designed with a Page Header, Page Footer, Page Form, and Report Details which print within the report detail area. The Report Details

should *not* have Surrounding Breaks. Break Headers and Break Footers are replaced by separate report Details.

The application accompanying this article was created with Clarion 6.1 9028; it contains all the report procedures from Part 1 of this article, and new procedures demonstrating similar capabilities utilising the BMC.

The C61Report1 procedure is a simple unformatted report with a Detail Header, Detail and Detail Footer, as in Figure 1.

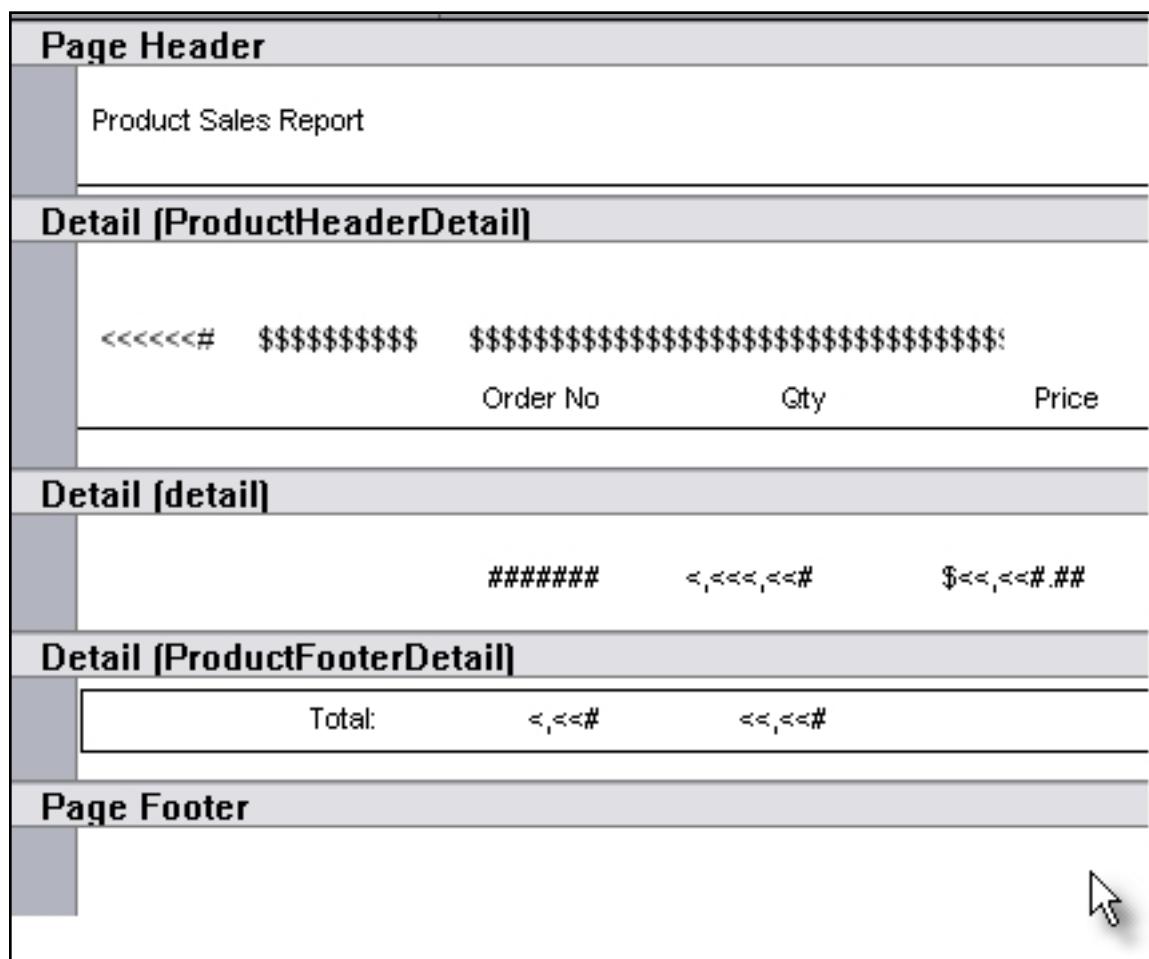


Figure 1. A simple report

I gave the ProductHeaderDetail and ProductFooterDetail details Use variables, as in Figure 2.

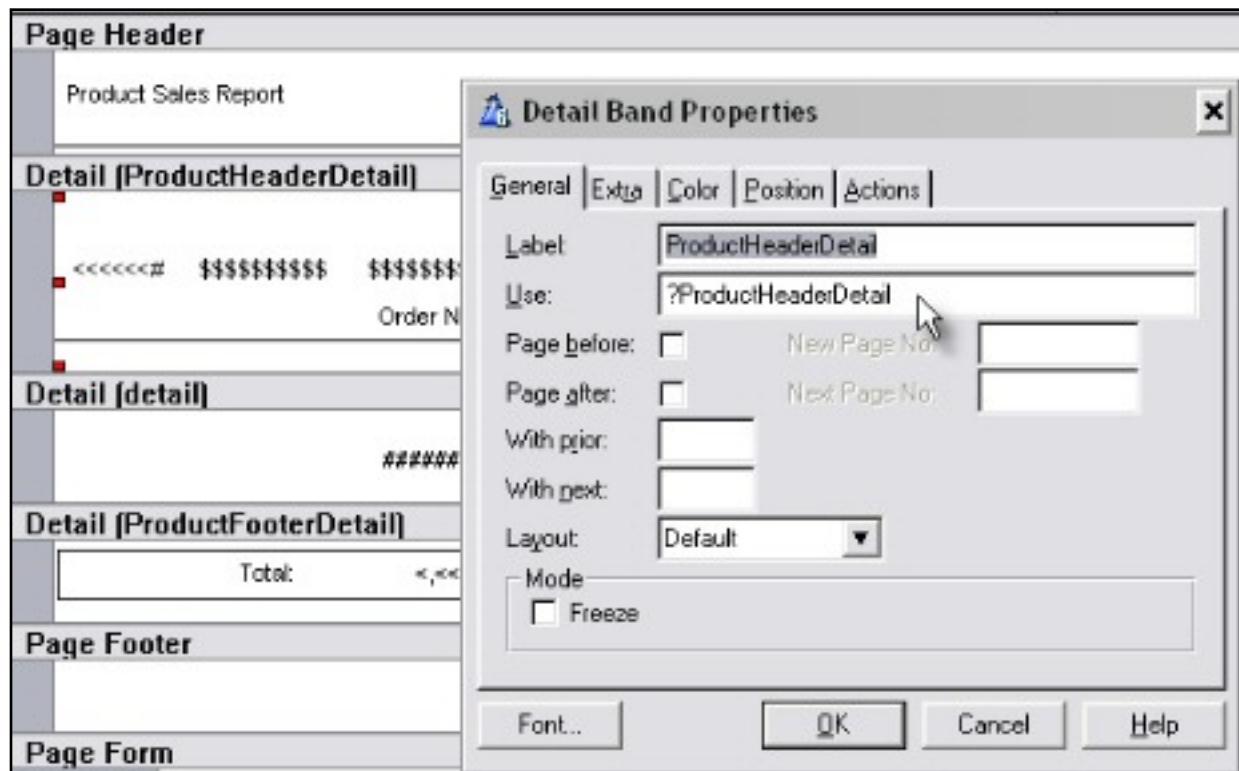


Figure 2. Adding use variables ([view full size image](#))

The Total fields in ProductFooterDetail are simply local variables. Their Total Type is set to None as in Figure 3.

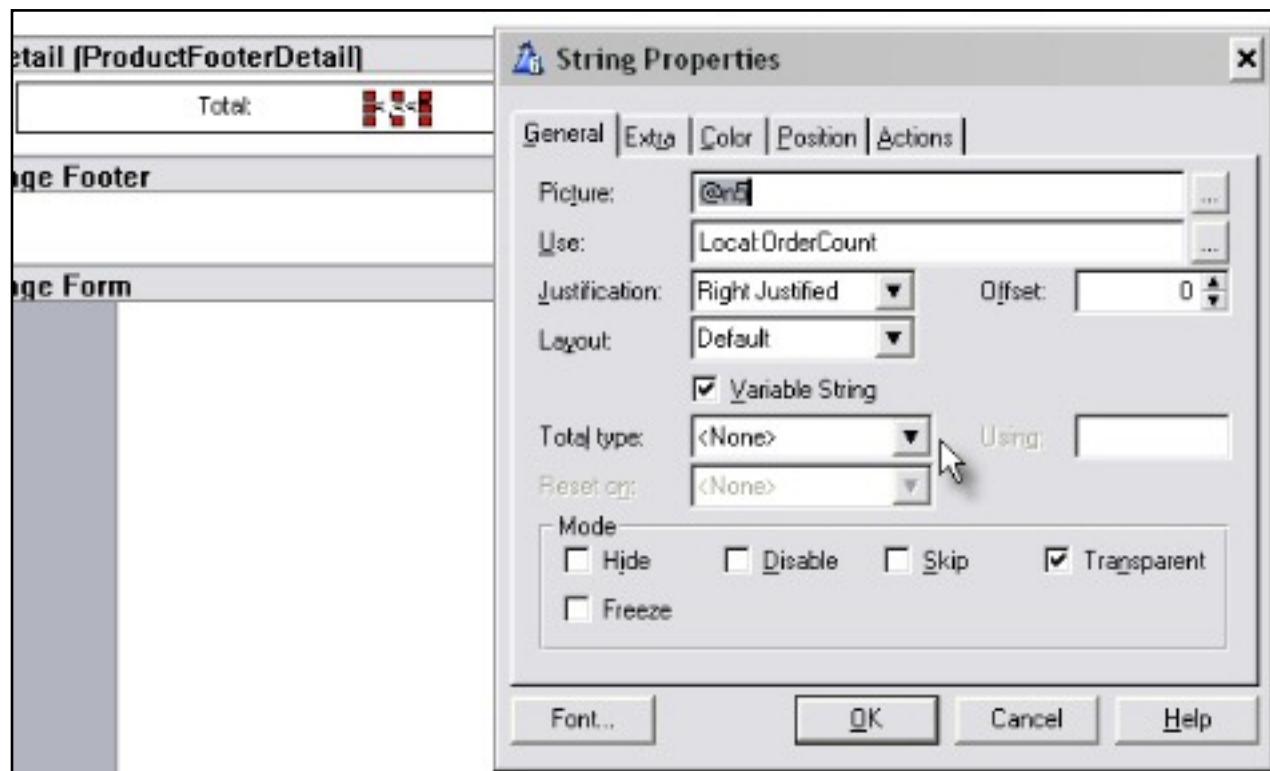


Figure 3. Setting the Total Type to <None> ([view full size image](#))

From the report properties button I set the filters for the Header and Footer Details to False so the report template won't print them – see Figure 4.

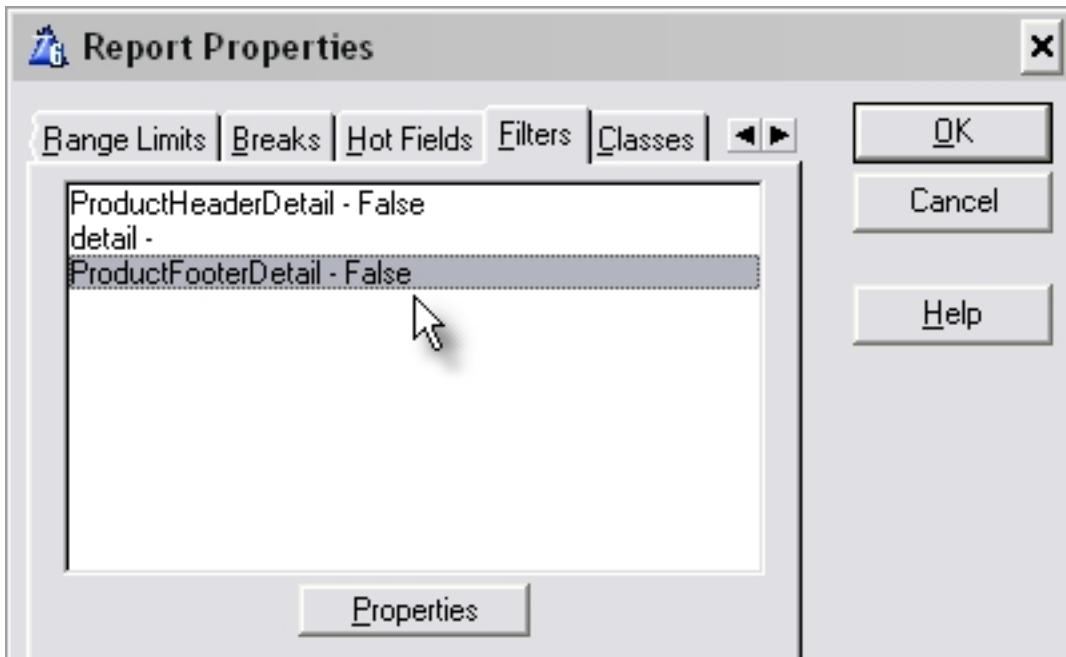


Figure 4. Turning off detail printing

When I run the report at this stage the Report Detail is all that prints, as expected (see Figure 5).

Product Sales Report				
0000001	5	\$7.50	37.50	
0000000	12	\$14.75	177.00	
0000001	5	\$7.50	37.50	
0000001	100	\$7.50	750.00	
0000001	4	\$28.75	115.00	

Figure 5. Printing the detail (view full size image)

Now its time to start using the BMC!

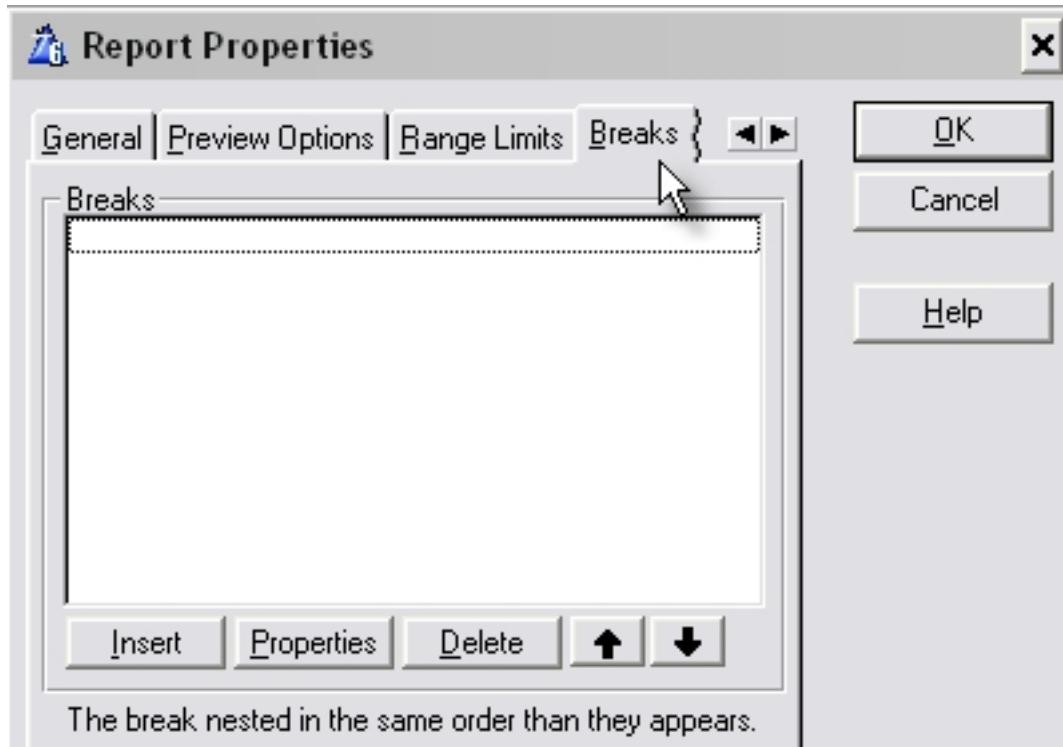
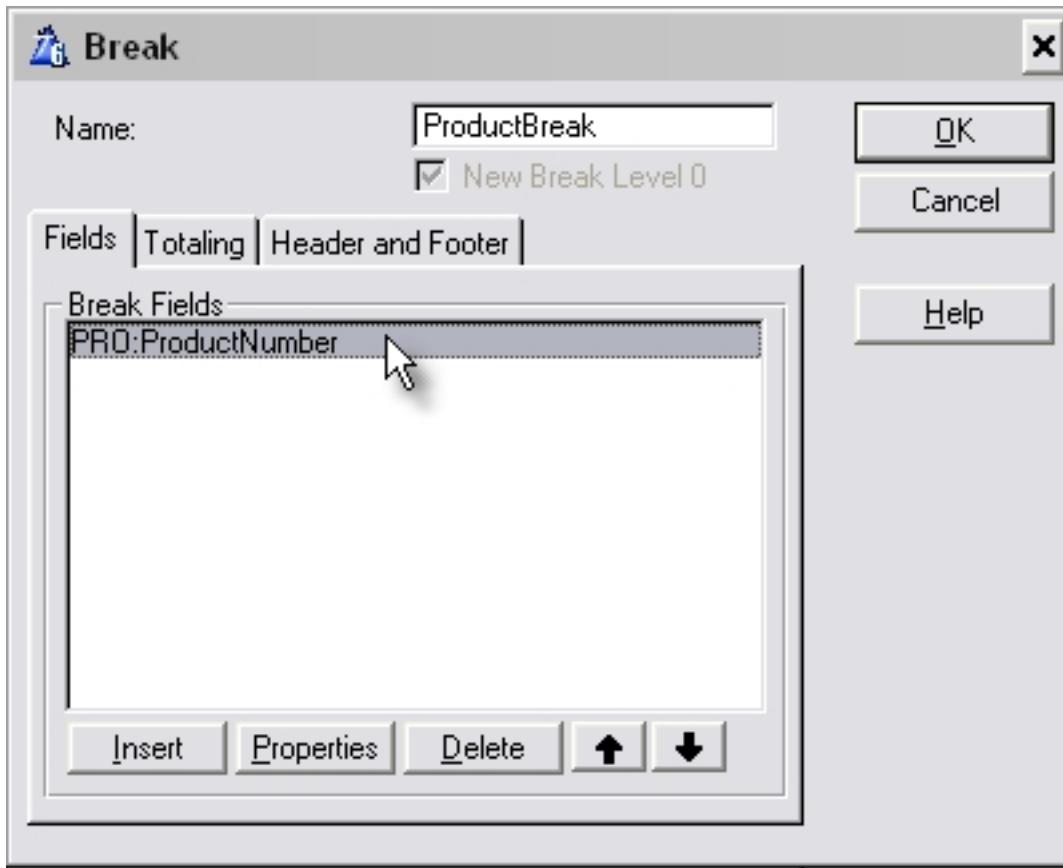
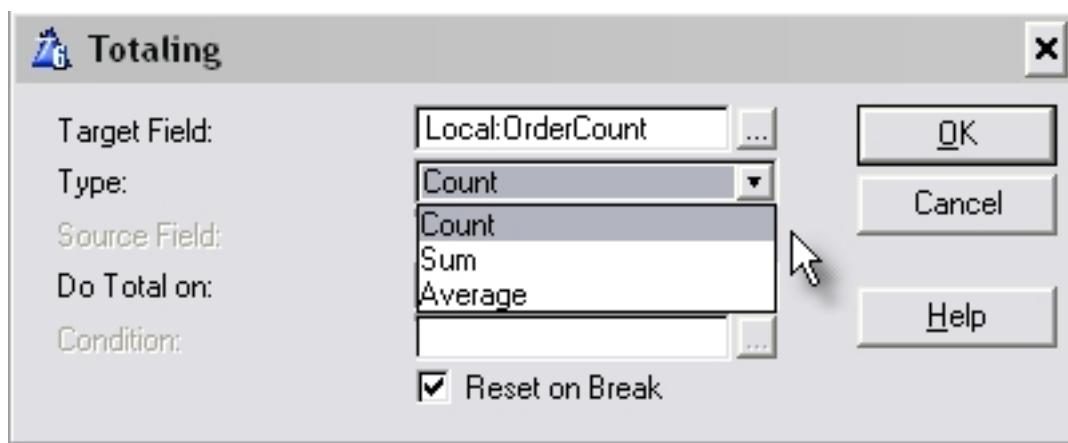


Figure 6. Using the new Breaks tab

The BMC is implemented from the Report Properties button, on the new Breaks tab. To specify a break, press the Insert button. The window that appears has three tabs. The first, Fields, requires a break Name and at least one Break Field. If multiple fields are selected, this break will be triggered when *any* of the specified fields changes. This flexibility makes the BMC technique more powerful than the break handling in traditional reports.

**Figure 7. Adding a break**

Since I have totals in the report, I need to specify some totalling for this report break. So I select the Totaling tab and insert the names of my local variables into the Target Fields to store the totals.

**Figure 8. Setting up totaling**

Total Types are limited to – as of this writing, at least – Count, Sum and Average. The traditional break handling also has Maximum, Minimum and

Page. When I select a Count type each record is counted, and the Source Field is disabled, otherwise I need to select a Source Field as the basis for the Sum or Average.

Totalling can be performed on All Records read, or conditionally. The total can be reset when the break is triggered. Leave the Reset on Break checkbox unchecked for report grand totals. If you need to perform a Page Total you will need to use the traditional totalling method, which won't usually be a problem since you would normally place these totals in the page footer.

In my example I have specified three total fields for this break, as shown in Figure 9.

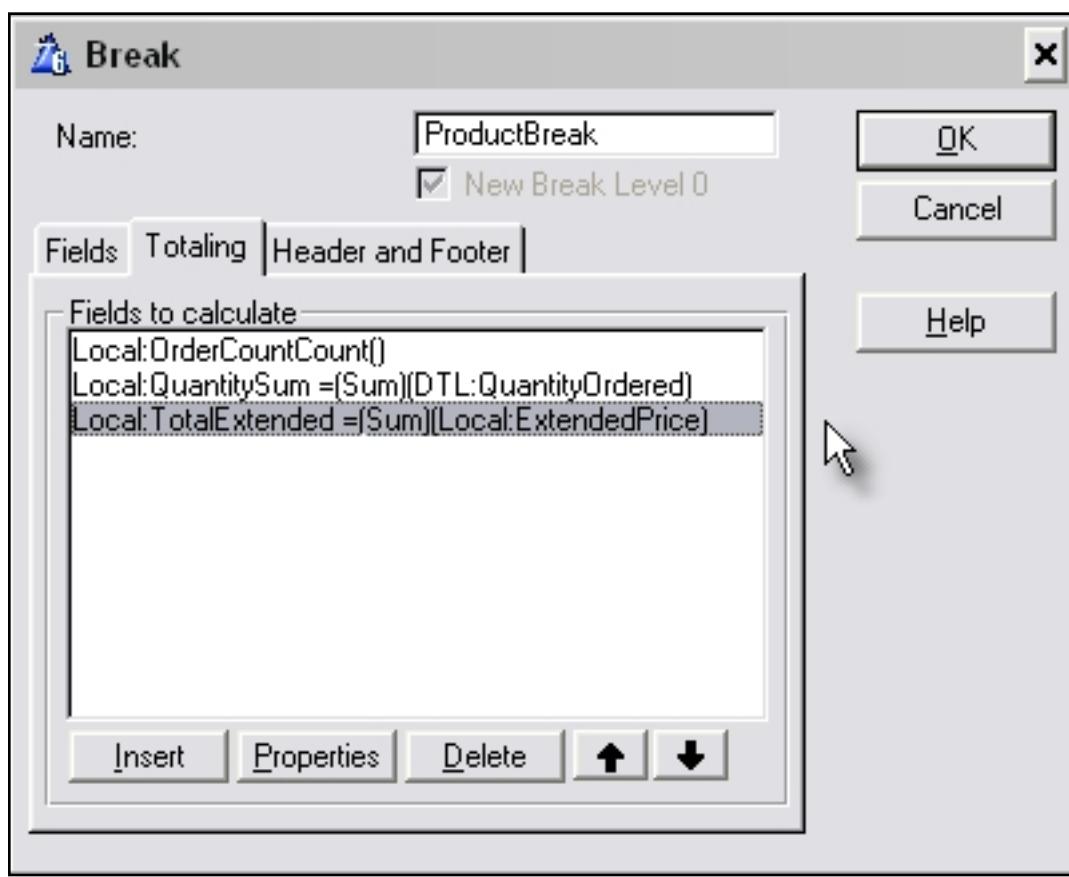


Figure 9. Three total fields in a break

The final tab for the BMC is Header and Footer where I specify which Header and Footer should be printed when a break is triggered.

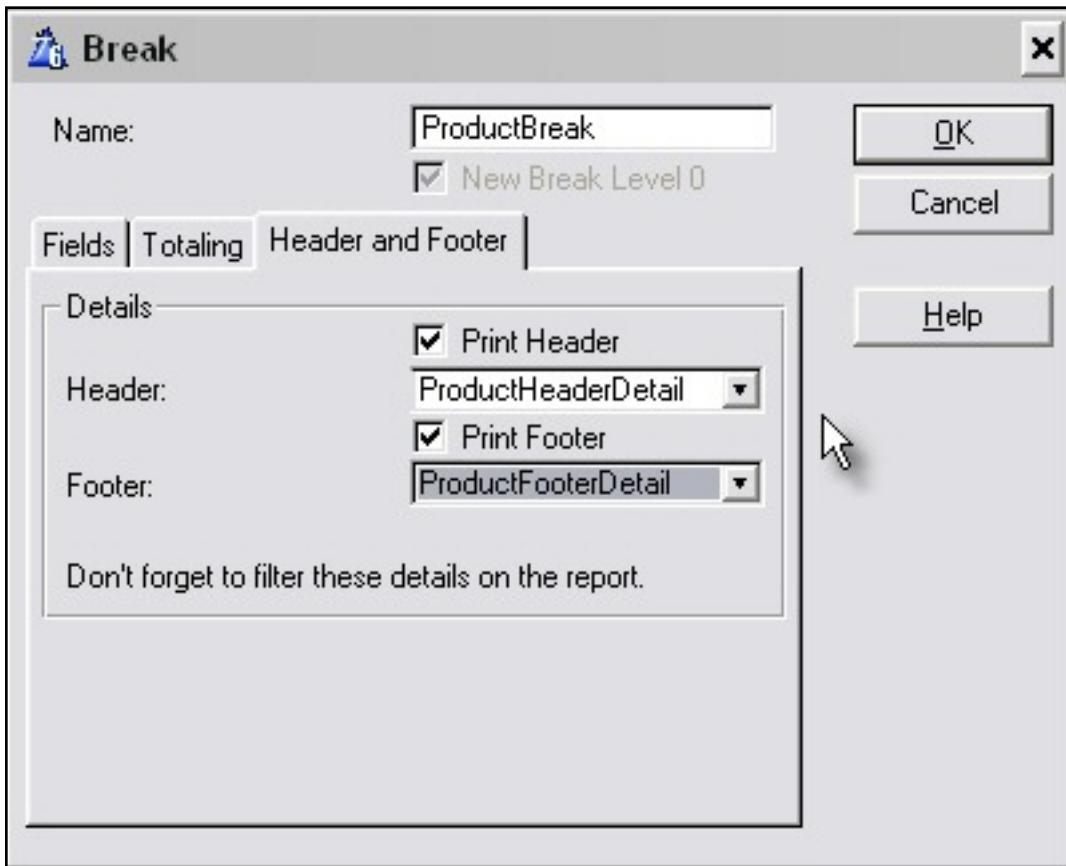


Figure 10. Printing headers and footers

At this stage I want both my Header and Footer to be printed, so I check both check boxes, as in Figure 10.

Formula Fields

So far this has all been pretty straightforward, so here is the first trap of the BMC. In Clarion 5 and earlier there was only one TakeRecord embed and it was always easy to advise developers "if you want something to happen in a Clarion report, place your code in the TakeRecord embed!" In Clarion 6.1 there are two TakeRecord embeds, one for the ProcessManager class, and one for the WindowManager class, so you might need to be careful which one you choose. In my example procedure I want to calculate an extended total and the traditional embed for performing this calculation,

`ProcessManager.TakeRecord (ThisReport.TakeRecord)` *doesn't work – see Figure 11.*

Product Sales Report

18	AL1234	Allium Flower			
		Order No	Qty	Price	Extended
		0000001	5	\$7.50	37.50
	Total:		1	5	\$0.00

Figure 11. Totals fail when the wrong embed is used ([view full size image](#))

The extended total gets calculated for each record, but the break footer totals are out of synch. So, where should I place my code? A clue for this can be found by using the infamous Formula Editor. At Devcon a show of hands indicated that only about 50% of attendees use the Formula Editor, and the rest of us were surprised that the number was so high! In my opinion the Formula Editor is much improved in C6, and it can be useful, as I am about to demonstrate.

Using the Formula Editor in C6.1 I found a new Embed Class for reports Before Processing Each Record. This embed wasn't there in C6.0, I know because I looked while writing this article! But this is the one I need – see Figure 12.

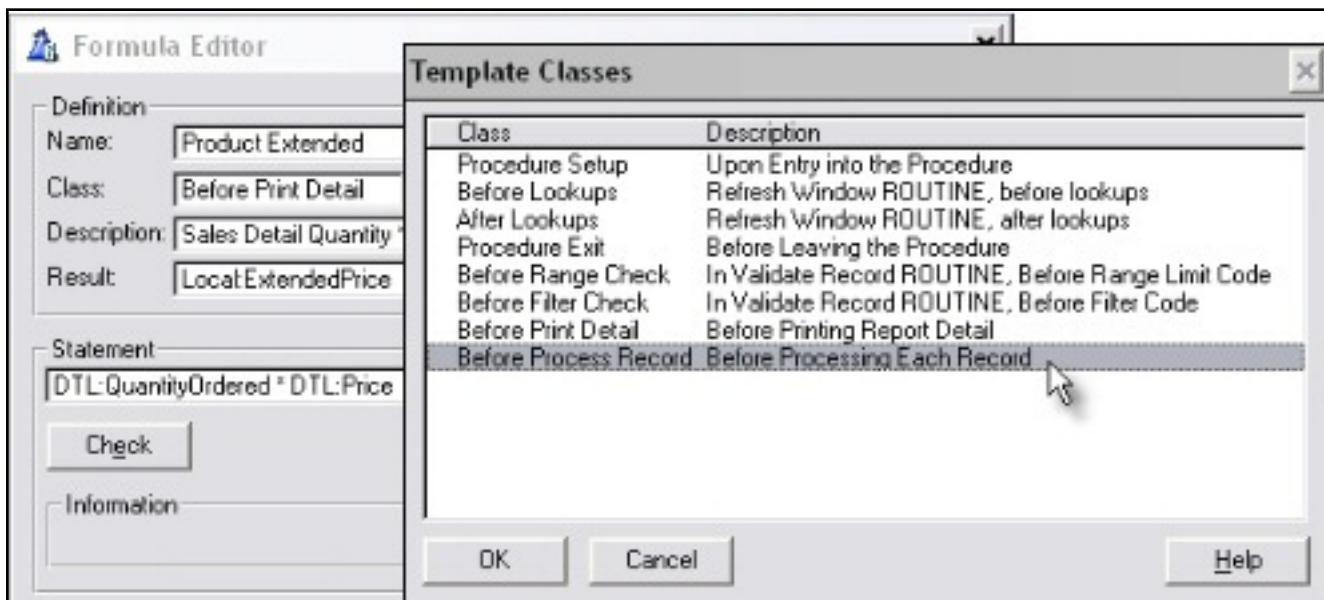


Figure 12. Viewing embeds in the Formula Editor ([view full size image](#))

I can see that this formula has worked correctly (see Figure 13), but where was the calculation inserted?

Product Sales Report				
Line No.	Order No.	Item Description	Extended	
			Qty	Price
18	AL1234	Allium Flower		
	0000001		5	\$7.50
				37.50
	Total:		5	\$37.50
20	AN329DLO	Antherium (2 Gallon)		
	0000003		12	\$14.75
				177.00
	Total:		12	\$177.00

Figure 13. Totals work! ([view full size image](#))

In the C61Report1 example procedure I have marked the correct embed point with a comment, and you can find it yourself by looking to see where the Formula Editor has placed the calculation. If you thought it would be the

WindowManager.TakeRecord embed (ThisWindow.Takerecord) you were right! So, the first trick to using the BMC, if you are used to typing TakeRecord to find your favourite embed point in Clarion reports, is to also press Ctrl-Enter to take you to the second TakeRecord embed!

That seems like a lot of work to get us no further than the traditional Clarion reporting techniques; what else is on offer? Well, one of my objectives was to format the Header, so where do I put the code for that?

C61Report2 procedure

If you look at the embed tree for a BMC report you will find a whole new set of embed points, starting with the Break Manager (BreakManagerClass) object in Figure 14.

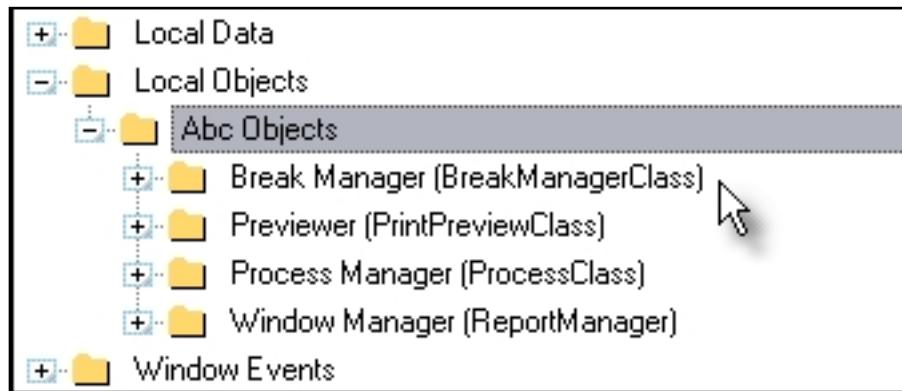
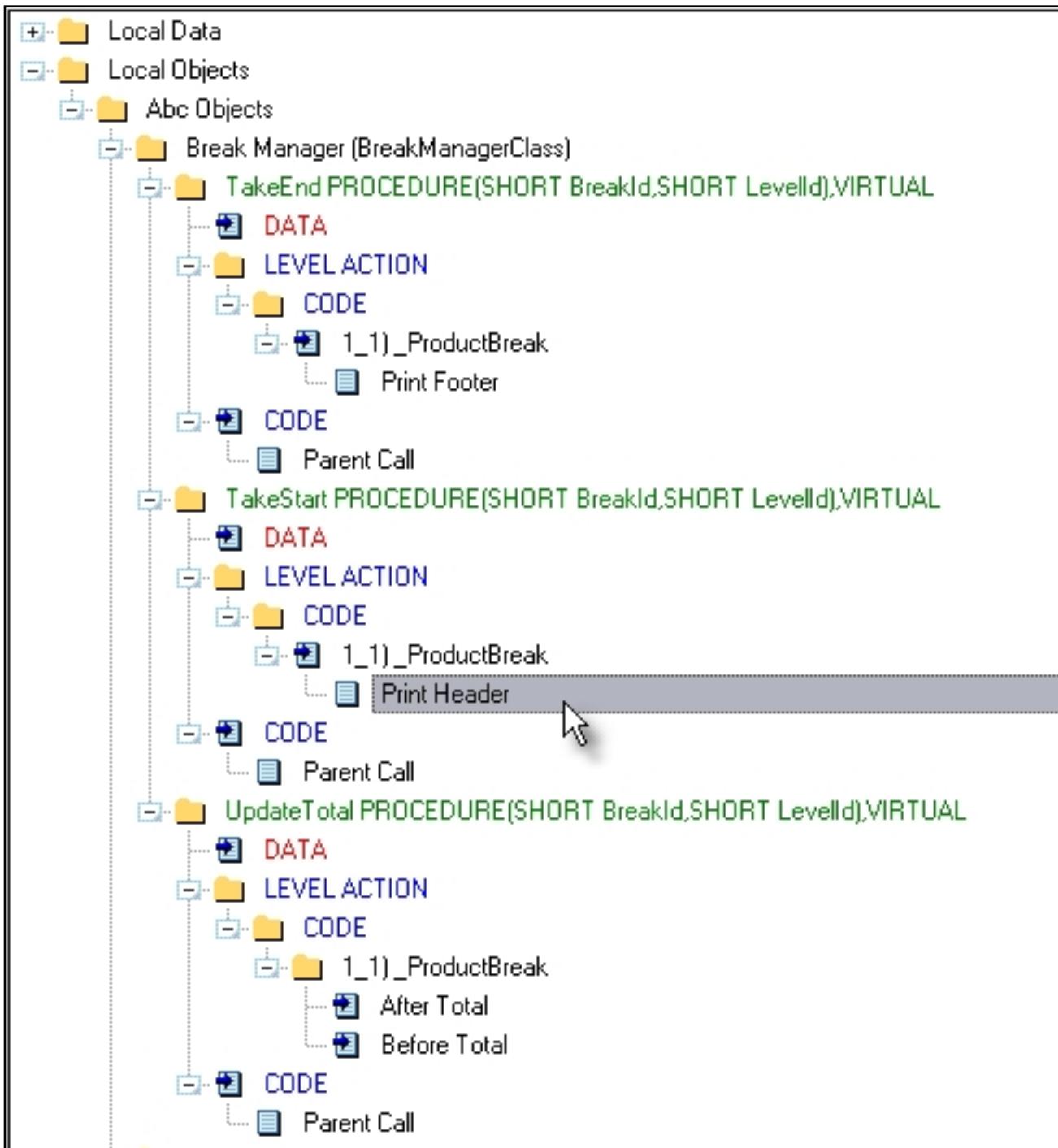


Figure 14. The Break Manager embeds

Expand this class embeds and you'll find some very useful sounding embed points, as you can see in Figure 15. Congratulations to Softvelocity for actually giving the embed points meaningful names; it makes selection of the appropriate embed point child's play!

**Figure 15**

I want to color the product name field blue in the ReportDetailHeader band for any product name beginning with the letter "B". The Print Header embed shown in Figure 15 seems like the right place, so let's see.

I place exactly the same code as used in previous versions of Clarion in this embed, before the PrintHeader.

```

SETTARGET(Report)
IF SUB(PRO:Description,1,1) = 'B'
    ?PRO:Description{Prop:FONT,3} = Color:Blue
ELSE
    ?PRO:Description{Prop:FONT,3} = Color:BLACK
END
SETTARGET

```

What happens when I run the report?

14	BB-16489	Bottle Brush Tree			
		Order No	Qty	Price	Extended
		0000001	5	\$28.00	140.00
		Total:	1	5	\$140.00
2	CP-57169	Cabbage Palm (4 Foot)			
		Order No	Qty	Price	Extended

Figure 16. Conditionally coloring text (view full size image)

It works!

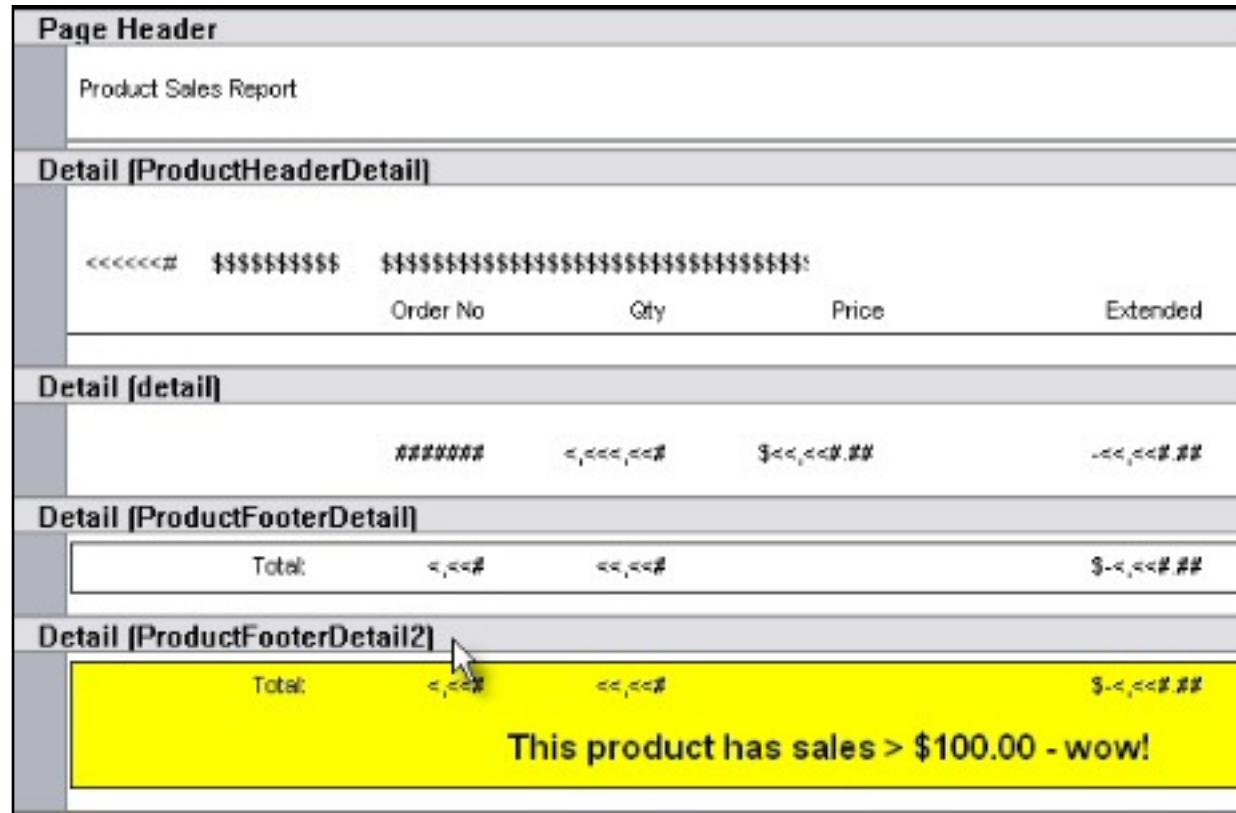
Performing Calculations on Break Footer Totals

In the previous article I conditionally colored a box in the DetailFooter yellow if sales exceeded \$100.00. See the code in C5xReport5 to see how I did this using the traditional reporting techniques. The technique I described is a little bit counter-intuitive, but it works. Does the BMC offer better ways of using report total fields more easily?

C6Report4 procedure

The traditional reporting techniques allow for only one footer per break variable. The BMC class has no such restriction, so I let this power go to my

head and created a second break footer – ProductFooterDetail2. I did this by copying and pasting the definition for ProductDetailFooter in the report source editor, and then I returned to the formatter to make the changes shown in Figure 17. I then set the filter for this detail to False, as described previously (see Figure 4).



The screenshot shows the Report Designer interface with the following structure:

- Page Header:** Contains the text "Product Sales Report".
- Detail [ProductHeaderDetail]:** A table with columns: Order No, Qty, Price, and Extended.
- Detail [detail]:** A table with columns: #, Order No, Qty, Price, and Extended.
- Detail [ProductFooterDetail]:** A table with columns: Total, Order No, Qty, and Extended.
- Detail [ProductFooterDetail2]:** A table with columns: Total, Order No, Qty, and Extended. This section is highlighted with a yellow background and contains the text "This product has sales > \$100.00 - wow!".

Figure 17. Multiple footers ([view full size image](#))

My intention here is to conditionally print ProductFooterDetail2 whenever the *total* sales for a product exceed \$100.00. So, I need to know what the totals sales for the product is, and then conditionally print the footer.

Since I am taking control of printing the footer, I need to remove the automatic printing of the footer that is handled by the BMC. I do this from the BMC properties window by un-checking the checkbox for the footer.

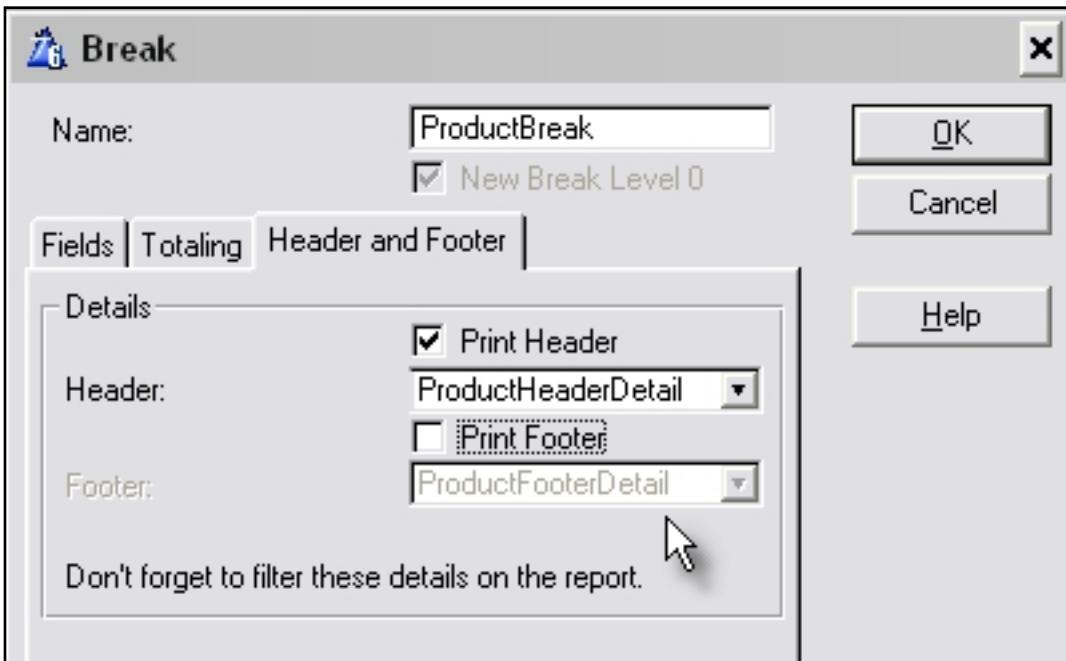


Figure 18. Disabling automatic footer printing

Now I need to embed the code to print the appropriate footer. The appropriate embed point would appear to be Print Footer. Since I have disabled the printing of the footer by the BMC it doesn't matter whether I put the code before or after the embed point - Figure 19.

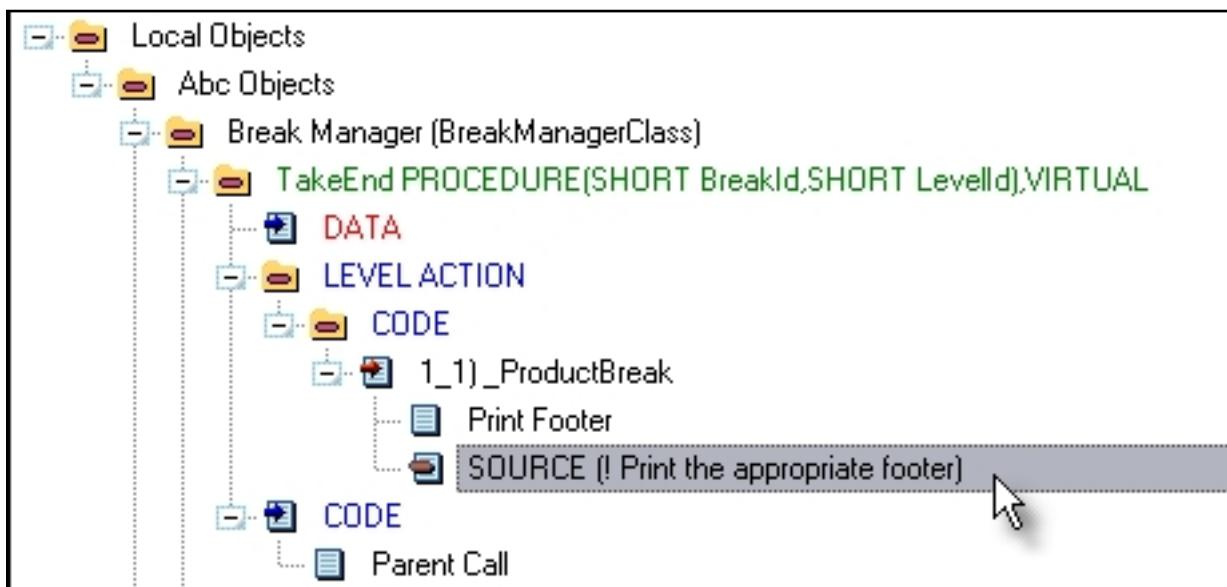


Figure 19. The Print Footer embed

I inserted the following code.

```

! Print the appropriate footer conditional on
! Total Extended sales for a product
IF Local:TotalExtended > 100.00
    PRINT(Rpt:ProductFooterDetail2)      ! Print excited footer
ELSE
    PRINT(Rpt:ProductFooterDetail)      ! Print normal footer
END

```

When I ran the report, it worked! See Figure 20.

Product Sales Report - C61Report4				
ID	Order No	Product Details		
		Qty	Price	Extended
18	AL1234	Allium Flower		
	0000001	5	\$7.50	37.50
	Total:	1	5	\$37.50
20	AN329DLO	Antherium (2 Gallon)		
	0000003	12	\$14.75	177.00
	Total:	1	12	\$177.00
This product has sales > \$100.00 - wow!				

Figure 20. Printing one of multiple footers ([view full size image](#))

If you don't already know, let me say that doing this with traditional Clarion reporting techniques would be quite difficult. I would need to change the height of the footer, resize the box, color the box, unhide my string field, and then change everything back again, for each product. The BMC has given us some meaningful embed points to do the things that developers, and our customers, expect our reports to do.

I'm sold!

How can I reprint break headers at the top of the page?

In my previous article I demonstrated the use of `WithPrior` and `WithNext` to help control widows and orphans in reports. The procedure `C5xReport5` demonstrates this with a traditional report, and procedure `C61Report5` shows exactly the same technique can be used with the BMC to produce the same outcome. Unfortunately Page two of `C61Report5` still doesn't indicate which product is being printed, as in Figure 21.

Product Sales Report - C61Report5				
	Order No	Qty	Price	Extended
	0000001	5	\$20.25	101.25
	0000001	10	\$8.00	80.00
	0000001	200	\$1.80	360.00
	0000002	20	\$2.50	50.00
Total:	7	344		\$997.25
This product has sales > \$100.00 - wow!				
24	BAYL15646	Bay Leaf		
Order No	Qty	Price	Extended	
0000001	12	\$2.50	30.00	

Figure 21. Missing headers at the top of the page ([view full size image](#))

I explained in the previous article how the Report Engine takes control of page overflow when using the traditional reporting technique of Surrounding Breaks, or `WithPrior` and `WithNext`, so it is impossible to know when a new page has been printed. Since the BMC gives us control over the printing of each detail band, I was curious to know if it was possible to enhance the printing of break headers, and specifically to reprint break headers at the top of a page.

C61Report6 procedure

I decided to experiment with the BMC and manually print the page breaks on this report.

If you look at the source code for a BMC report you will see a `PRINT()` statement for every detail in the report. I thought it would be possible to call a procedure every time a `PRINT()` statement was made that would check to see if I needed to print a new page. If I needed a new page I would issue an `ENDPAGE` and then print the `ProductHeaderDetail` at the top of the new page. The first thing I needed to do was set the height of each detail to a fixed height, rather than Default. Then I removed the settings for `WithPrior` and `WithNext`. Finally, I would need a variable to keep track of where I was on the page, `LastPosition`.

The procedure worked but it occurred to me that the code in the procedure was actually more appropriately implemented at the point where the `PRINT()` statement was made, since the BMC template generates these embed points. If this worked, maybe it would be possible to build my code into the BMC template?

Here is an example of the code I use *prior* to printing a detail. Similar code is used in the other embed points and can be seen in the `C61Report6` procedure.

```

! Code to print a new page if there is
! not room for a Detail
1. SETTARGET(Report)
2. LastPosition += ?Detail{Prop:Height}
3. IF LastPosition > Report{Prop:Height}
4.   LastPosition = ?ProductHeaderDetail{Prop:Height} +?Detail{Prop:Height}
5.   ENDPAGE(Report)
6.   PRINT(Rpt:ProductHeaderDetail)
7. END
8. SETTARGET()

```

Line 1 issues a SETTARGET because I want to use the properties of report

controls.

Line 2 updates a local variable, `LastPosition`, which is a LONG I am using to keep track of where I am within the report detail printable area. It sets `LastPosition` to the current value, plus the height of the detail band, since I am about to print a detail band. I use `?Detail{Prop:Height}` rather than the actual fixed height value of the detail band because I might change the height of the band and I don't want to have to change my code.

Line 3 checks to see if the last position is greater than the height of the detail's printable area. If it is the Report Engine will execute a page break. I don't want the Report Engine to do this; I want to do it myself! So, in Line 4 I reset `LastPosition` to the height of my `ProductHeaderDetail` + the height of the detail band. I do this because in Line 5 I start a new page with `ENDPAGE`. Then on Line 6, I manually print a `ProductHeaderDetail`, and I know the template is about to `PRINT(Rpt:Detail)`.

The following figures show the result of this code.

5	B-45483	Bamboo (3 Gallon)			
		Order No	Qty	Price	Extended
		0000001	4	\$20.25	81.00
		0000001	5	\$25.00	125.00
		0000002	100	\$2.00	200.00

Page: 1 

Figure 22. Before the page break ([view full size image](#))

The bottom of Page 1 looks okay, but did I print the page break, or did the Report Engine do it?

Product Sales Report - C61Report6

5	B-45483	Bamboo (3 Gallon)			
		Order No	Qty	Price	Extended
		0000001	5	\$20.25	101.25
		0000001	10	\$8.00	80.00
		0000001	200	\$1.00	360.00
		0000002	20	\$2.50	50.00
	Total:	7	344		\$997.25
	This product has sales > \$100.00 - wow!				

24	BAYL15646	Bay Leaf			
		Order No	Qty	Price	Extended

Figure 23. After the page break ([view full size image](#))

Figure 23, the top of Page 2, indicates that I did it, because the ProductHeaderDetail is there, at the top of the page!

Is this solution perfect? Unfortunately not! The top of Page 4 Figure 24 shows that there is a (minor?) problem with orphans. Ideally I would have liked at least one detail to be printed at the top of the page, with the footer. But I can live with this solution.

39	R782093	Rose			
		Order No	Qty	Price	Extended
	Total:	1	12		\$30.00

Figure 24. The orphan problem ([view full size image](#))

I have also included, just for interest, the `C61Report`⁷ procedure which uses a series of strings to display the value of `LastPosition` as the report prints. I used this during testing as a debugging technique.

In conclusion

These two articles have compared the handling of Report Breaks using the traditional reporting techniques in Clarion with the new `BreakManagerClass` available in Clarion 6.1. It is my belief that, over time, the `BreakManagerClass` will eventually replace the traditional reporting techniques for the vast majority of Clarion users. The reasons for this belief are simple; the `BreakManagerClass` concepts are easy to understand, the embed points are meaningfully labelled and therefore easy to use, the embed points allow a developer to do what needs to be done simply, and the embed points provide the opportunity to do things that are simply not possible with the traditional reports.

I provided a demonstration of a technique for managing page overflow, which I believe could, and should, be built into these templates. I said at the beginning of this article that these templates are evolving, I hope this is correct and that Softvelocity continues to develop this powerful tool.

[Download the source](#)

Reader Comments

[Add a comment](#)

Hello i'm printing a report from direrents QUEUE's and...

Copyright © 1999-2004 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.



Quick Links

[HOME](#)

[Subscribe/Renew](#)

[Clarion Books](#)

[FAQ \(Frequently Asked Questions\)](#)

[Log In](#)

[Download PDFs](#)

[Free Membership](#)

[Free Articles](#)

[Advertise](#)

[Reader Comments](#)

[Write For ClarionMag](#)

[Privacy](#)

[Contact](#)

Clarion Magazine

Tackling Global Queues in Clarion 6

by John Seganakis

Published 2004-11-26

Clarion 6 has introduced a new threading model which has definite benefits, but problems that were once easily solved with a global memory queue in earlier preemptive model cause problems in the new cooperative threading model. Many articles have been dedicated to this subject, but no general solution has appeared.

Global queues are very useful for loading memory with data that is commonly used throughout an application, especially when there is a large data set or if the data is acquired through an intensive I/O process. I have many apps that use memory queues to share data globally. And I prefer to convert existing applications to the preemptive threading model as easily as possible.

The ideal situation is to generalize the common problem for any queue. My solution is to encapsulate a queue in a class, and use generic access methods to manage the queue's data. I call this class QueueManagerClass, and in this article I'll describe how it works and how you can use it to solve queue management problems in Clarion 6.

The QueueManagerClass has a queue reference property:

```
ManagedQueue &Queue, Protected
```

ManagedQueue as the protected attribute, so it can only be accessed by this class or a derived class. This ensures that the queue's data access will be properly synchronized with all threads. The other essential property is

```
QueueCS &ICriticalSection,Protected
```

which is the critical section needed for thread synchronization. All access to the queue directly must call the Wait and Release methods of the critical section so memory will not become corrupt.

It would have been nice to use QueueManagerClass methods with names matching the familiar queue functions, but some just cannot be defined in a class. The result is a compiler error suggesting that you are redefining a basic definition. This is actually a mistake on the part of the compiler, as the scope for the method in the class scope is different from the global scope. Perhaps in time this bug will get repaired. So I used method names common in the ABC file access classes.

Fetch is of course used in place of Get, and Insert for Add. The definition for the Insert method is as follows

```
QueueManagerClass.Insert PROCEDURE(*Queue q)
  code
  self.Lock()
  if self.Init()
    self.ManagedQueue = q
    ADD(self.ManagedQueue)
    if self.SyncThreadQueue
      Add(q)
    end
  end
  self.Unlock()
```

When inserting I pass a queue which is defined locally on the thread, so all my existing code using the old global queue need not change. The new code is as readable as the old code – I just make sure I use the QueueManagerClass functions for all reading/writing.

In the Insert method, the Queue Manager calls the Lock() function that in turn calls the Clarion Wait() function in the critical section. This maintains thread synchronization, so only one thread will be accessing the memory queue, preventing memory corruption. The Init() function merely asserts

that a queue has been initialized for management. Typically this is performed as follows:

```

MyQueue      Queue, Type
SomeStr       string(20)
SomeLong      long
End
QM           QueueManagerClass
MyQRef       &MyQueue
...
MyQRef  &= New MyQueue
QM.Init(MyQRef)

```

There is no need to dispose of memory as the queue manager will do it when the object is destroyed in the destructor. There is an optional parameter (QueueManagerClass.Init PROCEDURE(*Queue q, byte bDispose=true) where bDispose can be set to false if the caller prefers to free memory. This option would be used in the following case, where Clarion will automatically destroy the memory allocated for the queue:

```

MyQueue      Queue
SomeStr       string(20)
SomeLong      long
End
QM           QueueManagerClass
...
QM.Init(MyQRef, false)

```

Here is typical queue usage using the standard Clarion approach:

```

Free(MyQ)
Loop i=1 to 10
  Clear(MyQ)
  MyQ.SomeStr = 'Item ' & i
  MyQ.SomeLong = i * 100
  Add(MyQ)
End

```

And here is the same code using global queue management:

```
QM.Free()
Loop i=1 to 10
    Clear(MyQ)
    MyQ.SomeStr = 'Item ' & i
    MyQ.SomeLong = i * 100
    QM.Insert(MyQ)
End
```

Only the access and retrieval functions need to be changed.

Alternatively the entire queue can be copied to a queue declared in the thread. I've also created the function GetQueue() that does that. Primarily I created it to use with list boxes. The queue contained within the class can be passed to a list box, which may be running on multiple threads, but it is not very wise to pass off the reference without clearly knowing what the control will do with it, as this increases the likelihood of memory corruption. It is much safer to have a copy of the queue for each thread.

General Access Functions

Also within the class are general access functions, where the actual structure of the queue need not be defined within the code scope. That is, once the queue manager has been initialized with a queue, other code can operate on that queue without knowing its exact structure. Each queue record can be accessed by queue and field index; however thread synchronization must be kept in mind and handled manually if multiple threads are accessing the queue. The class methods Lock() and Unlock() allow access to the critical section contained in the class.

The names of all fields in the queue can be listed in a string, using the QueueManager's Who method:

```
QM.Lock()
Loop I=1 to QM.Fields()
```

```

    FieldText = FieldText & QM.Who(i) & '<9>'
End
QM.Unlock()

```

The data can be accessed, using the QueueManager's What method:

```

QM.Lock()
Loop I=1 to QM.Records()
    QM.Fetch(i)
    Loop j=1 to QM.Fields()
        FieldData = QM.What()
        ...
    End
End
QM.Unlock()
Data in the queue can also be set:
QM.Free()
QM.Lock()
Loop I=1 to QM.Fields()
    ...
    QM.Set(I, FieldData)
    QM.Insert()
End
QM.Unlock()

```

Data in the queue can also be set:

```

QM.Free()
QM.Lock()
Loop I=1 to QM.Fields()
    ...
    QM.Set(I, FieldData)
    QM.Insert()
End
QM.Unlock()

```

The Set method will also take the name of the field passed as a string as well as an integer value:

```

QM.Free()
QM.Lock()
...
QM.Set('SomeStr', FieldData)
...

```

```
QM.Set('SomeLong', FieldData)
QM.Insert()
QM.Unlock()
```

As I indicated earlier, this generic approach has specific advantages. You could create a procedure that takes a QueueManagerClass reference as a parameter, where the procedure doesn't know the queue's actual structure. This is quite different from passing a general queue reference, since the class provides complete access to the queue it contains

For example suppose some application defines two QueueManagerClass instances. One contains a reference to a queue with this structure:

```
Q1    Queue
rValue      real
SysTime     long
Description  string
Position    long
...
End
```

and the other a queue with this structure:

```
Q2    Queue
rValue      real
SysTime     long
Dsc        string
Heading    string
...
End
```

As a simplistic example suppose a DLL is created with a generic procedure to square the value in the queue and record the clock cycle this occurs. The procedure could be defined as follows:

```
SquareSysValue Procedure(*QueueManagerClass pQM)
rValue real
nRecords long
I long
Code
nRecords = pQM.Records()
```

```
Loop I=1 to nRecords
    PQM.Fetch(i)
    rValue = PQM.What('rValue')
    rValue = rValue * rValue
    PQM.Set('rValue', rValue)
    PQM.Set('SysTime', clock())
end
```

Note here that when the **What** method is called with a string parameter, **What** returns the value of the corresponding field.

This hypothetical DLL would not require the declaration of either **Q1** or **Q2**, and would work as long as the referenced queues contained fields with the expected names. If not, the error object would be set in the class, which could be checked. (The code snippet provided is very simple and does no error checking.)

The sample project includes a test procedure which demonstrates updates on multiple threads. Select **Test|Prepare Test** to open three windows with list boxes browsing the global queue (see Figure 1). The global queue was placed in a separate DLL for the demonstration.

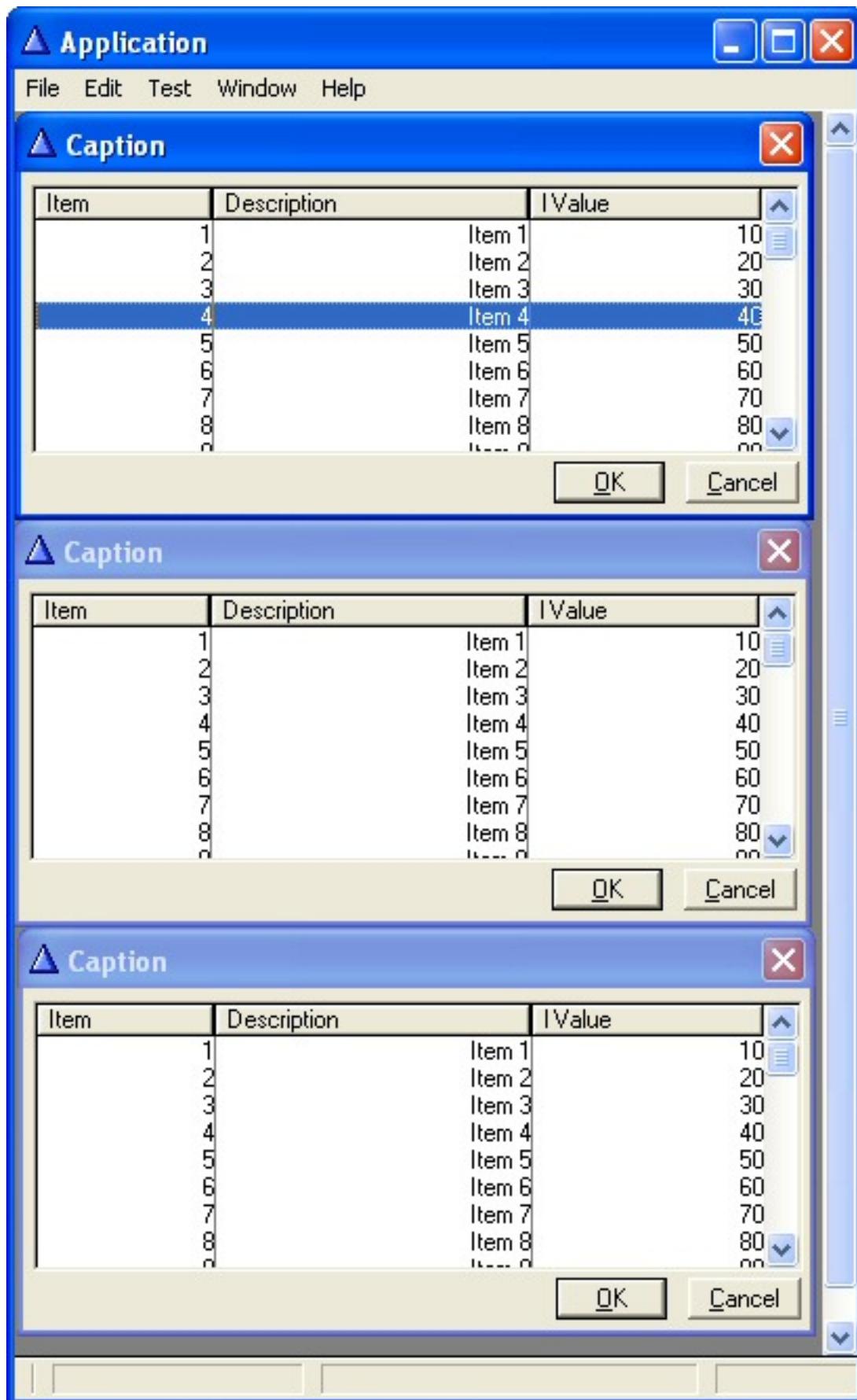
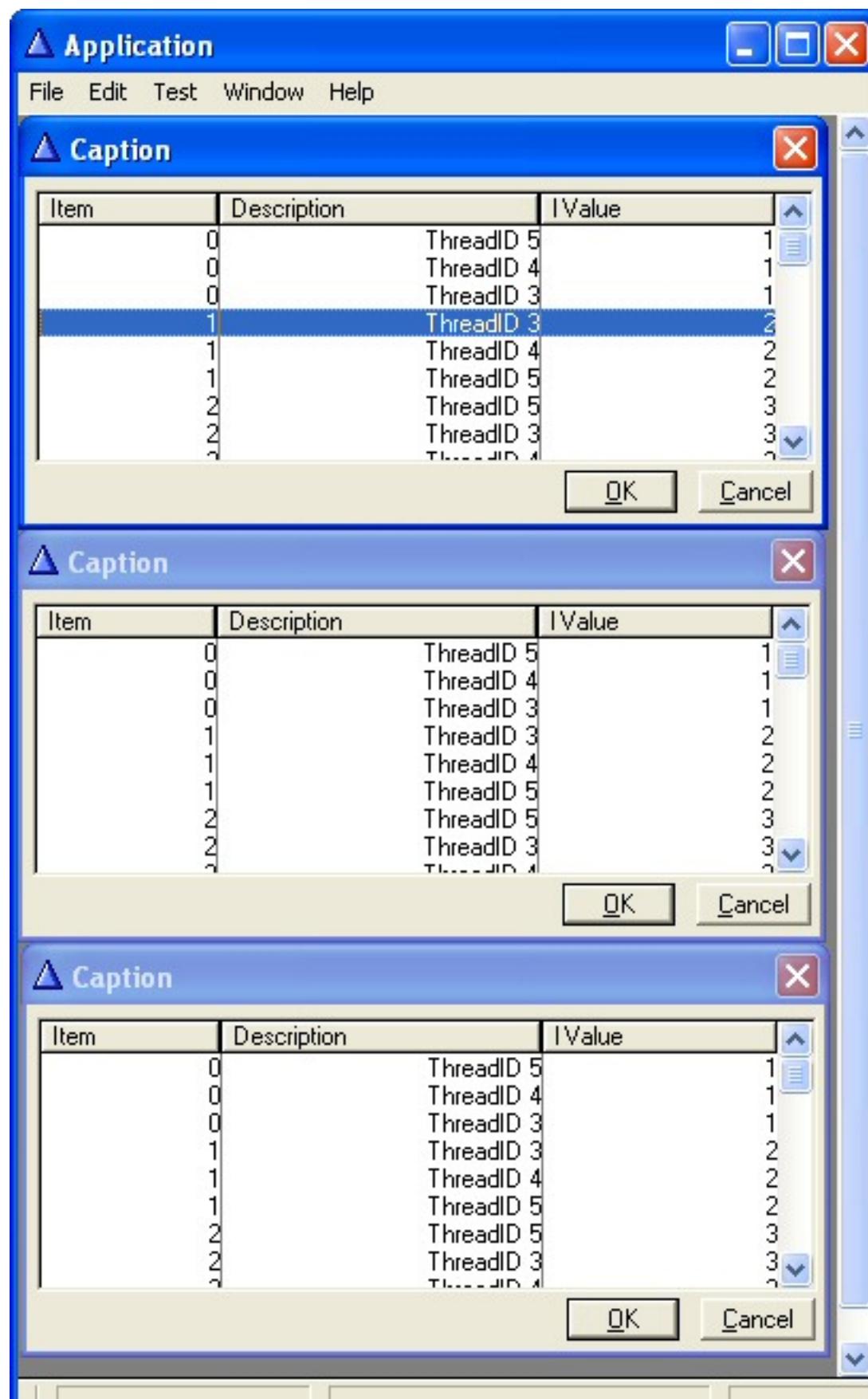


Figure 1. Sample application with three test browse windows open on multiple threads for updating

To start the test Select Test|Start, which starts a timer where each thread updates the queue for two seconds. Once complete the browses will refresh (see Figure 2).



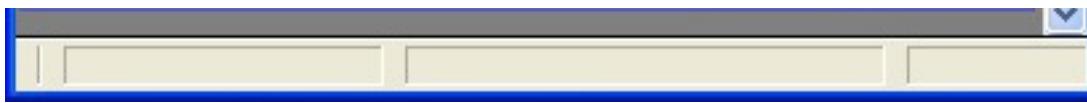


Figure 2. Browse windows after the test completes

In Figure 2 the Item field is the item number each thread is adding, the description contains the thread id of each thread adding the item, and elapsed time is displayed in the lValue field. As you can see, each thread is updating the queue within the same clock cycle and each thread is adding its item to the queue.

Summary

Global queues are useful in application wide uses to avoid loading large amounts of data multiple times or loading data from I/O intensive methods. The QueueManagerClass easily allows the use of global queues in the cooperative threading model in Clarion 6. I also found this class useful previous to Clarion 6, as it is easy to achieve generic queue access by passing the managed queue, within a class, to a procedure..

QueueManagerClass Reference

QueueManagerClass contains a queue for thread synchronization. This allows an easy transition from global queues to Clarion 6; with very few code changes a global queue can be used in the Clarion 6 cooperative thread model. Additionally, generic queue access is readily available and the error object is used to return errors.

Usage

Typically existing queue declarations are changed to a type declaration.

Example:

Assume a global queue is defined as:

```

MyQueue      Queue
SomeStr       string(20)
SomeLong      long
End

```

In a module the code usage may be

```

Free (MyQueue)
MyQueue.SomeString = 'FirstItem'
MyQueue.SomeLong = 1
Add(MyQueue)
Get(MyQueue, 1)

```

First, make the global queue a type:

```

TYPE:MyQueue      Queue, Type
SomeStr          string(20)
SomeLong         long
End

```

Now change the module declarations to include the following

```

MyQueue      TYPE:MyQueue
QM          &QueueManagerClass

```

The code using a global queue would change thusly

```

QM.Free()
MyQueue.SomeString = 'FirstItem'
MyQueue.SomeLong = 1
QM.Insert(MyQueue)
QM.Fetch(MyQueue, 1)

```

Properties

ManagedQueue &Queue, Protected

Reference to the main instance of a queue to be managed. It is not accessible.

QueueCS &ICriticalSection, Protected

The critical section to assure only one thread can update or read the queue at a time.

SyncThreadQueue Byte

The property to keep a copied queue in sync with the main managed queue. Set to true to have the queues in sync. Defaults to false.

Methods

ClearBuffer()¹

Same as Clear() as operated on a queue.

Delete(*Queue q)

Deletes the item previously retrieved in the managed queue.

```
QM.Fetch(1)
QM.Delete()
```

Delete(*Queue q)

Deletes the specified item contained in the passed queue from the managed queue.

```
QM.Fetch(MyQueue,1)
QM.Delete(MyQueue)
```

Fetch(Unsigned nRow)

Fetch is used like the Get function. This will fill the managed queues buffer with the specific data in the row requested.

```
QM.Fetch(1)
```

SortQueue(? k1)

Fetch is used like the Get function. This will fill the buffer of the passed queue with the specific data in the row requested.

```
QM.Fetch(MyQueue,1)
```

Fields(),long

Returns the number of fields contained in the managed queue. Assuming the managed queue is defined as the previous definition of MyQueue, QM.Fields() returns 2

Free()

Frees the managed queue. Same as the free function.

```
QM.Free()
```

GetQueue(*Queue q)

Will return the entire contents of the managed queue to the passed queue

Init(), byte

Returns true if the class contains a managed queue; false otherwise

Init(*Queue q, byte bDispose=true)

Initializes the QueueManagerClass with the queue to manage. Once the

queue to be managed is declared and passed to Init there is no need to directly access it. The reference to the queue passed is stored in the class. bDispose is set to true to automatically dispose of the allocated queue, or to false to manually control memory disposal or if Clarion is releasing memory.

Example:

```

MyQueue    Queue, Type
SomeStr        string(20)
SomeLong       long
End
QueueReference  &MyQueue
QM            QueueManagerClass
Code
QueueReference = New MyQueue
QM.Init(ManagedQueue)

```

Now the class may be passed as a reference to other procedures for queue access.

Insert()¹

Inserts items entered into the managed queues buffer. Same functionality as Add.

```
QM.Insert()
```

Insert(Unsigned nRow)¹

Inserts an item at the position specified from the buffer of the queue passed. Provides the same functionality as Add(Queue, pointer).

```
QM.Insert(i)
```

Insert(*Queue q)

Inserts items into the managed queue from the buffer of the queue passed.

```
QM.Insert(MyQueue)
```

Insert(*Queue q, Unsigned nRow)

Inserts an item at the position specified from the buffer of the passed queue.

```
QM.Insert(MyQueue, i)
```

Records()

Returns the number of records contained in the managed queue.

```
nRecords long
I long
nRecords = QM.Records()
loop I=1 to nRecords
    QM.Fetch(MyQueue, i)
    ! - perform operation with queue item ---
end
```

Set(String Field, ? QueueData)¹

Sets the field specified of the queues buffer with the data passed in QueueData. The Field parameter can be an integer value or the string value of the queue record to change

```
MyQueue    Queue, Type
1      SomeStr           string(20)
2      SomeLong          long
End
```

Where the numerical value of each record is denoted by the numbers on the left, always numbered from the top down.

```
QM.Set('SomeLong', 1)
```

similarly,

```
QM.Set(2, 1)
```

SortQueue(? k1)

Sorts the queue according to the specified sort items passed

```
QM.SortQueue(' +SomeLong, -SomeString' )
```

Update()¹

Updates the managed queue previously retrieved item; similar to Put functionality.

```
QM.Fetch(1)
QM.Set('SomeString', 'Primary Value')
```

QM.Update()

Updates the managed queue from the buffer of the passed queue. Provides Put functionality.

```
QM.Fetch(MyQueue,1)
MyQueue.SomeString = 'Primary Value'
QM.Update(MyQueue)
```

What(String Field), ?¹

Returns the specified field in the managed queue, where each member in the queue is indexed from the top, after the queue declaration, down starting at 1 for the top item. The Field parameter can be an integer value or the string value of the queue record.

```
MyQueue      Queue, Type
SomeStr          string(20) ! Field 1
SomeLong         long       ! Field 2
End
MyQueue.SomeString = 'FirstItem'
MyQueue.SomeLong = 1
QM.Insert(MyQueue)
QM.What(2), returns 1
```

Who(Unsigned nCol), STRING¹

Returns the name of the field numbered from top, after the queue declaration, down starting at 1 for the top item.

```

MyQueue      Queue, Type
SomeStr       string(20) ! Field 1
SomeLong      long        ! Field 2
End
MyQueue.SomeString = 'FirstItem'
MyQueue.SomeLong = 1
QM.Insert(MyQueue)
QM.Who(2) , returns 'SomeLong'
```

¹ Usage of this method should call the synchronization methods Lock() and Unlock(), if multiple threads are accessing the queue. Refer to the examples in General Access Functions.

[Download the source](#)

John Seganakis has been a Software Engineer for about 12 years professionally, and has programmed in Clarion, C++, VB, PASCAL, BASIC, and some assembler.

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 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.





Quick Links

[HOME](#)

[Subscribe/Renew](#)

[Clarion Books](#)

[FAQ \(Frequently Asked Questions\)](#)

[Log In](#)

[Download PDFs](#)

[Free Membership](#)

[Free Articles](#)

[Advertise](#)

[Reader Comments](#)

[Write For ClarionMag](#)

[Privacy](#)

[Contact](#)

Clarion Magazine

Clarion News

[Search the news archive](#)

Clarion Developers Challenge Week 11 Winner

First place for Week 11 of the Clarion Developers Challenge goes to Ed Schneider. Ed finished first in last week's contest with a very solid 13 correct picks. Four other players finished a close second with 12 correct picks. For finishing first in this last week's Clarion Developers Challenge, Ed won a copy of gReg, which was donated by Jesus Moreno and Gitano Software. However, Ed wishes to throw his winnings back into the pot for this week's Clarion Developers Challenge winner!

Posted Monday, November 29, 2004

Office Inside 1.58

Office Inside 1.58 available (including demo exe) is now available. This release handles the annoying "A program is trying to automatically send email..." message that Outlook displays every 2 seconds.

Posted Monday, November 29, 2004

SB5 "IP Data Server" Script

An updated "SoftVelocity IP Data Server.sbi" include script for SetupBuilder 5 to distribute the SoftVelocity IP Data Server will be available on Monday. The updated script makes it possible to automatically uninstall the IP Data Server (stop/remove service).

Posted Monday, November 29, 2004

FM3 3.50 Beta Adds Sybase

File Manager 3 v3.50 Beta has just been uploaded to the web. This is the first release allowing you to test the brand new Sybase support. Please be aware that it is early stages, but all feedback and bug reports are welcome. A tip for

anyone using Sybase through ODBC: The default driver setting Adapative Server Anywhere 9.0. If you would like to change this, set LOC:ASAODBC = 'You ASA Driver' at the top of the Login routine in the connect procedure. Please note, this must use the new Connect Control Template. See the Example App for more info.

Posted Monday, November 29, 2004

Simsoft Christmas Stocking

Until 1 December 2004 you can get the Simsoft Christmas Stocking for just 129\$US (normal price \$196). This year's stocking is filled with: Simsoft Templates (normal price \$69US); Simshape Templates (normal price \$49US); Simpad Templates (normal price \$49US); SimTabTree Template (normal price \$29).

Posted Monday, November 29, 2004

ClarionShop Big Prize Giveaway

Requesting your free ClarionShop catalog will automatically enter you in the draw for the biggest prize in Clarion Programming history. One developer will win all present and all future Clarion accessories from a number of top suppliers. The draw date has been set for 3 January 2005 so make sure that you order your catalog before then.

Posted Monday, November 29, 2004

New Product: cwAdvantage

iAlchemy has announced the upcoming release of its new product cwAdvantage, a database manager utility designed for the AdvantageDB product from www.extendedsystems.com. cwAdvantage imports TXDs, converts the TXD to SQL/DDL, optionally creates the Advantage database, and in the future will execute DDL.

Posted Monday, November 29, 2004

xWordCOM Library 1.3

xWordCOM Library v1.3 is now available. New in this version: MergeDocuments method was fixed - template file is now closed

automatically after merging; Demonstration program updated to show different variants of documents merging.

Posted Monday, November 29, 2004

IP Driver and Data Server Now Shipping

SoftVelocity has begun shipping the IP Driver and Data Server (IPDS). The product is being e-delivered.

Posted Wednesday, November 17, 2004

xRuntimeStyle Manager v1.8

New in this version of xRuntimeStyle Manager: xRunTimeStyle Manager Class changed for compatibility with Clarion 6.1 (Build 9029); Updated documentation, demonstration program and installation kits for Clarion 5.5 and Clarion 6 are available.

Posted Wednesday, November 17, 2004

New DCT2SQL Template

Changes to DCT2SQL include a new Dct2Firebird00 template. This template *could* be useful if you have already done a first time generation, read back the dictionary with ODBC, everything is prefixed correctly, and you need to generate the creation DDL script again. Your mileage might vary. This is a work in process, and the FOREIGN KEY generation might not be quite right.

Posted Wednesday, November 17, 2004

Clarion Developers Challenge Week 9 Winner

First place for Week 9 of the Clarion Developers Challenge goes to David Bowness. David finished this week with a solid 12 correct picks, edging out Alejandro Contreras with 10 correct picks and four other players who finished a close third with 9 correct picks. For finishing first in this week's Clarion Developers Challenge, David will be receiving IMPEX, donated by Mike McLoughlin and Sterling Data. Thanks, Mike McLoughlin and Sterling Data for your generous support to the Clarion Developers Challenge and the Clarion community.

Posted Wednesday, November 17, 2004

Sterling Data Named Third-party Vendor of the Week

Mike McLoughlin and Sterling Data have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week. Mike is awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of IMPEX, the drag-n-drop data import/export templates. Once again, Mike McLoughlin and Sterling Data have shown their support for the Clarion community, and we'd like to take this opportunity to thank them.

Posted Wednesday, November 17, 2004

New Super Stuff and Super Invoice Released

Super Stuff 6.54 is now available. This release has a new Classes tab in the ResizeGlobal template, which you can use to override the base class used for mhResizeFrame, or to specify an alternate default resizing class for all of your windows. This release also fixes a problem resizing windows with toolbars. Super Invoice 6.52 now allows non-child EIP fields. You're responsible for managing these fields in the SetQueueRecord and UpdateBuffer methods. See the embed tree for a bit more guidance. WARNING: This is an "advanced" feature, and there is little/no hand-holding, so beginner beware. There will be an example of this in an upcoming version. Several bugs in SuperInvoice were also fixed. For all purchase and upgrade issues (including passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (952) 745-4941.

Posted Monday, November 08, 2004

vuFileTools Trend Micro Issue Resolved

Bill Roe, the author of vuFileTools, has tracked down the reason why PC Cillin reports vuFT2.DLL as a virus. Any common call to GetCurrentProcess() (a standard Windows function that has been around since Win32 days) causes the calling program to be interpreted by Trend Micro as a virus. In all of the 70 odd functions (including internal subroutines and private procedures and

functions) available within vuFileTools, commenting out this single call eliminated the Trend Micro virus fiasco. The latest release of vuFileTools uses a workaround for this problem and no longer triggers the virus warning.

Posted Monday, November 08, 2004

ADDA 1.0.5

ADDA (Advanced Data Dictionary Architect) version 1.0.5 is released. Bug fixes include: Stored procedures & views weren't saved; In some cases, nationalized strings were not preserved when recreating tables. Enhancements include new "Other Entities" category, which allows entering unrestricted SQL queries such as user defined types, functions, or any type of customization. ADDA is a multipurpose toolkit for designing, creating and maintaining the database layer throughout the entire application lifecycle.

Posted Monday, November 08, 2004

WinEvent 3.24

WinEvent 3.23 has incomplete export files for cw55 and cw50. Please download WinEvent 3.24.

Posted Monday, November 08, 2004

Berthume Software Named Third-party Vendor of the Week

Greg Berthume and Berthume Software have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week. Greg is awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of cpTracker Pro, the complete Contact, Project and Content Management package! Once again, Greg Berthume and Berthume Software have shown their support for the Clarion community, and we'd like to take this opportunity to thank them.

Posted Monday, November 08, 2004

BLAT 1.3

Big Legacy to ABC Tamer Ver 1.3 is now available. This fixes a regression in ver 1.2 for "priority" numbers for converting %ControlEventHandling embeds.

(Legacy "Post" and "Pre")

Posted Monday, November 08, 2004

Icetips Xplore 1.205

Icetips Xplore version 1.205 is now available. This fixes a nasty bug that caused some serious bloating of INI files where Xplore was saving/restoring the browse format.

Posted Monday, November 08, 2004

XPTHEME 2 For Clarion 6.1

XPTHEME 2 for Clarion 6.1 adds XP theme capabilities to your Clarion 6.1 application, when running on Windows XP.

Posted Monday, November 01, 2004

EasyMultiTag 2.07 For C61 9029

EasyMultiTag ver 2.07 is ready for C61 9029. Free for all registered customers. EasyMultiTag will allow you considerably to expand functionality of the BrowseBox: Tagging of any records (both separate and ranges) using Windows-style, by colour, by icons, invisible, using hot keys and popup menu or buttons; High-speed filtration and calculation of the totals. Instantaneous Inversion; Transfer all necessary information about tagged records, Range Limits and Sort Order into standard procedures REPORT and PROCESS; Storing and restoring; Built-in support for Drag&Drop, EIP and QBE; and more. Price: \$79.

Posted Monday, November 01, 2004

PowerOffice Home Page Moved

The PowerOffice home page has moved - please update your links. Products at this site include XP-Taskpanel; OutlookBar; PowerXP-Theme.

Posted Monday, November 01, 2004

dpQuery 2.01

dpQuery 2.01 is now available. Fixed: Now you have full access to all wizard controls and window in the templates; Code generation and embedded points

in case of more then one dpQuery code templates in one procedure; Changed SetProp and OverrideProp methods; Updated demo application (shows how do you customize dpQuery Wizard window with your own wallpaper, prompts, font style, buttons etc); Added dpQuery Wizard, a new button to assign fields with the similar descriptions automatically; Added pQuery class documentation; New methods to create and work with the dynamic control - CreateDynControl and SetDynProp. Free upgrade for all registered customers. Price: \$149

Posted Monday, November 01, 2004

xDataBackup Manager Pro 2.0

Update xDataBackup Manager Pro v2.0 is now available. New in this version: C6 version compiled under Clarion 6.1 (Build 9026); Small bugfix re error with hanging program when xDataBackup tried backup of missing data files and keys fixed. Updated Demonstration program and Installation Kits for Clarion 6.1 (Build 9026), Clarion 5.5 and 5 are available.

Posted Monday, November 01, 2004

xDataBackup Manager Lite 2.0

xDataBackup Manager Lite v2.0 is now available. New in this version: C6 version compiled under Clarion 6.1 (Build 9026); Small bugfix re error with hanging program when xDataBackup tried backup of missing data files and keys fixed. Updated Demonstration program and Installation Kits for Clarion 6.1 (Build 9026), Clarion 5.5 and 5 are available.

Posted Monday, November 01, 2004

Advertising On Clarionfoundry

If you subscribe to Clarionfoundry, for \$20 per year, you get to advertise free on the article pages.

Posted Monday, November 01, 2004

dpQuery 2.00

dpQuery ver 2.00 has been released, and a new demo is available. dpQuery

is a tool which includes libraries and a template that adds a powerful interactive resource (Query wizard) for importing of practically any external data into your program. Price: \$149.

Posted Monday, November 01, 2004

Ingress ODBC Patch

CA has released a patch for ODBC. Sean Hennessey has tested this with Clarion and reports that it works.

Posted Monday, November 01, 2004

Riebens News Server

All Riebens products and 3'rd party tools have a news server where users can discuss the tools, register requests and also bugs.

Posted Monday, November 01, 2004

DOS Printer 9.05

DOS Printer is a C6.1 program written using (amongst others) CPCS Reports, PDF Tools, and VividHelp's EmailReport. It may especially be of interest to anyone still using CPD2.1. DOS Printer sits in the system tray and monitors for the presence of a certain text file. If that file exists, the file is opened and printed using Windows mechanisms to any Windows printer that you specify. It acts just like a "virtual printer driver". It can also create PDF files, and email your reports as PDF attachments or as RTF in the body of the email. New in this release is the ability to take a rich text file and use it as the background for a report, thus allowing you to use a "virtual letterhead" with colours, fonts and logos - all from within your DOS Program. This helps make the email function more useful as you no longer need physical letterhead paper.

Posted Monday, November 01, 2004

DomainWatchDog - 100% Clarion Code

A full release of DomainWatchDog and written Doc (PDF) files, is now available. The app is for managing and monitoring your domains (up to 65,000), as well as debugging them when something breaks. The app is

100% free when managing five or fewer domains, and the built-in debugging tool is free regardless. The application is 100% Clarion 5.5EE and uses only a single publicly available DLL for compression of WhoWas and DNSWas archives. The app also contains a complete HTTP server for using it's built in BetterWhois debug tools remotely, and BetterWhois also provide many unique low level tests such as DNS PING and EMail server testing, DNS tracing from ROOT, and much more.

Posted Monday, November 01, 2004

Capesoft In The UK

Capesoft will be carrying out product training at the next UK Clarion User Group meeting on Nov 22nd. New members always welcome!

Posted Monday, November 01, 2004

C6 Icon Extractor

An icon extractor for C6, but without any docs.

Posted Monday, November 01, 2004

Product Specific News Server

Ben Dell is offering dedicated news server channels to interested Clarion developers, for a small fee.

Posted Monday, November 01, 2004

Clarion Utilities \$49

For a limited time you can get your choice of gCal, gCalc, gQ, gNotes, gSec, and gFileFind for \$49 each (reg \$69 - \$149). The Look Good Package and gReg are also on sale for \$99 each (reg \$119 - \$299).

Posted Monday, November 01, 2004

Clarionfoundry RSS Feed

Clarionfoundry now has an RSS feed for site updates.

Posted Monday, November 01, 2004

Postgres 8.0 Beta 3 Dev 1 Released

Kelvin Chua points out this latest Postgres beta.

Posted Monday, November 01, 2004

Copyright © 1999-2004 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.



Quick Links

[HOME](#)

[Subscribe/Renew](#)

[Clarion Books](#)

[FAQ \(Frequently Asked Questions\)](#)

[Log In](#)

[Download PDFs](#)

[Free Membership](#)

[Free Articles](#)

[Advertise](#)

[Reader Comments](#)

[Write For ClarionMag](#)

[Privacy](#)

[Contact](#)