



# Clarion magazine

**Volume 1, Number 7 - August 1999**

## Issue Index

### [Tom Hebenstreit New Reviews Editor](#)

I'm pleased to announce that Tom Hebenstreit has joined Clarion Magazine as Reviews Editor and columnist. Many of you know Tom for his excellent work over the past two years coordinating, writing and editing reviews for Clarion OnLine.  
Posted on August 10, 1999

### [Working With Control Files II](#)

Not sure how to handle single-record control files in ABC? Nik Johnson adds a class and template to Steve Parker's control file handing scheme.  
Posted on August 10, 1999

### [Is It Six Months Already?](#)

It's already been half a year since Clarion Magazine first published! Here's a look at what's appeared in Clarion Magazine over the past six months, and a preview of what's to come, including DevCon '99 coverage.  
Posted on August 10, 1999

### [Updated COSP](#)

The Clarion Open Source Project pages are now available to the public! We've revamped the program to make it easier for interested parties to participate in open source development. And in case you missed the announcement in June, the Developers Open Source Public License is now finalized!  
Posted on August 10, 1999

### [New Open Source Products](#)

Jeff Slarve's BitList code (class & template), Pat O'Brien's file dialog wrapper (class and sample app), and Richard Sylkie's trace code (class & template) have been added to the list of products under the Developers Open Source Public License. This page is now available to the public!  
Posted on August 10, 1999

### [Product Review: Imaging Templates](#)

Tom Hebenstreit, Clarion Magazine's new Reviews Editor, gives NextAge Consulting's Imaging Templates the full treatment.  
Posted on August 17, 1999

### [Interview: College Aid Calculator](#)

Although Clarion as a development tool is little known, some of the products created with Clarion have gained wide acceptance. One of these is the College Aid Calculator. Clarion Magazine recently interviewed developer Steve Brown.  
Posted on August 17, 1999

### [The SQL Answer Cowboy](#)

He may be 40, but he hasn't fallen out of the saddle just yet. The SQL Answer Cowboy rides through town again.  
Posted on August 17, 1999

### [Undocumented Debugging](#)

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

If you're interested  
take our poll &  
let us know!

- Main Page
- Log In
- Subscribe
- Open Source
- Links
- Mailing Lists
- Advertising
- Submissions
- Contact Us
- Site Index
- ClarionMag FAQ
- Download PDFs
- Search ClarionMag

Undocumented functions - there's always a risk in using them because they might go away in the next release, but sometimes they're worth a look. Here are four undocumented Clarion functions that can help you debug your applications.

Posted on August 17, 1999

#### [Alias - Who Was That Masked File?](#)

"File alias" is one of those concepts that is at once extremely difficult and extremely simple. Steve Parker explains what file aliases are, and how and when they should be used.

Posted on August 24, 1999

#### [Open Source Products Update](#)

The list of third party products using the Developers Open Source Public License continues to grow! Just added: Chris Behling's all-Clarion graphing code, and Pierre Tremblay's slider (from an earlier ClarionMag article).

Posted on August 24, 1999

#### [The Clarion Advisor: Changing Dictionaries](#)

If you've ever tried to change an application's dictionary you know that you're asking for trouble if the new dictionary is different from the old one. Mike Pickus explains why, and provides a utility that allows you to safely change your application's dictionary.

Posted on August 31, 1999

#### [Larry Teames On Reports](#)

Larry Teames takes apart a Clarion report and teaches it how to print multi-page letters the right way.

Posted on August 31, 1999

#### [Propitious Memory Corruption](#)

Propitious WHAT?! David Bayliss examines the theory and practice of relational integrity and file aliases in Clarion 5/ABC, with comparisons to legacy Clarion.

Posted on August 31, 1999

#### [Clarion News - August 1999](#)

Your first stop for news of the Clarion development community.

Posted on August 31, 1999

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).



# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## Press Release

### Tom Hebenstreit Clarion Magazine's New Reviews Editor

I'm pleased to announce that Tom Hebenstreit has joined Clarion Magazine as Reviews Editor and columnist. Many of you know Tom for his excellent work over the past two years coordinating, writing and editing reviews for Clarion OnLine. I've been wanting to strengthen Clarion Magazine's reviews section for some time, so I'm particularly happy to welcome Tom to the staff. Tom has been a Clarionhead since the the CPD days, and has used every version of Clarion since then (yes, even CDD/CFD).



I have a theory that almost any Clarion programmer is either also a musician or was once into darkroom photography. I don't know how handy Tom is with a camera, but I can tell you that he does vocals, keyboard and guitar in Bill Mummy's band The Jenerators. Bill Mummy, as you may recall, played Will Robinson in the '60s TV show Lost in Space.

So until fame goes to Tom's head, he'll be reviewing the growing number of Clarion third party products. I think the third party market is healthier than it's ever been, and with Topspeed's work on the new OLE layer the number of accessories that can be used with Clarion will be increasing dramatically.

If you have a product you'd like to see reviewed, email Tom at [reviews@clarionmag.com](mailto:reviews@clarionmag.com). And if you have a web page you'd like to see in the [categorized site list](#), send an email to [links@clarionmag.com](mailto:links@clarionmag.com).

Dave Harms  
Publisher

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

If you're interested  
take our poll &  
let us know!

#### In This Issue

[Tom Hebenstreit  
New Reviews  
Editor](#)

Posted on August  
10, 1999

[Working With  
Control Files II](#)  
Posted on August  
10, 1999

[Is It Six Months  
Already?](#)  
Posted on August  
10, 1999

[Clarion News -  
August 1999](#)  
Posted on August  
10, 1999

[Updated COSP](#)  
Posted on August  
10, 1999

[New Open Source  
Products](#)  
Posted on August  
10, 1999

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).



# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## A Class And Wrapper For Handling Control Files

by Nik Johnson

I'm an inveterate template tinkerer. It's a mixed blessing. By the time I abandoned my CPD 2.1 model file for the Logix Project Manager, there was more logic devoted to branching conditionally around various options than to perform the actual task at hand. By the time CDD 3007 rolled around, I had so much invested in modifying Todd Carlson's templates that I didn't dare switch to newer Clarion templates. But while they lasted these modified tools made me a lot more productive than I would otherwise have been.

I'm still tinkering, but the introduction of OOP and the ABC library has meant that I can now work within the TopSpeed framework rather than around it. [Steve Parker's article](#) on control files in the previous issue provides an opportune way to illustrate the convenience and power of OOP/ABC while at the same time extending my toolkit.

### Defining The Problem

What I want to build is a "set and forget" method of handling control files. As Steve points out, some file structures require a SET/NEXT approach, others do better with GET/POINTER. Still others may require other access methods. I want to spend an absolute minimum of time and effort incorporating control files into future applications, independent of the file system.

Steve has enumerated the things our tools need to be able to do:

- Open the file
- If the file is missing, optionally create it
- Read the file's only record in a way appropriate to the file structure
- If the file has no records, add one
- Update the record
- Close the file

The following narrative mimics the way I go about building tools for my own use. First, list the things that have to be done; second, establish a general design; third, build a skeleton; fourth, fill in the details. This works for a small shop, but if you're building tools for sale or working in a larger organization you may want to skew this pattern toward heavier documentation of the design in advance of coding.

### This Wheel Has Already Been Invented ... Almost

The ABC FileManager class provides facilities to accomplish five of these six tasks, so it makes sense to start with that as a basis. A class based on FileManager can inherit all of its functionality and minimize the new work that needs to be done.

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

If you're interested  
take our poll &  
let us know!

```
MyFileManager CLASS(FileManager)
Fetch          PROCEDURE, BYTE, PROC
                END
```

The new method provides a place to add Steve's access logic, but it doesn't include provision for specifying which version of that logic to use. ABC classes are insensitive to driver, but in this case that luxury is not available.

Two possibilities come to mind. First, the fetch algorithm could accept a flag to designate how access to the one and only record in the file is to be gained. A byte should be sufficient, since more than 255 variations on the get-only-record theme are unlikely. However, if the file driver changes, it could be inconvenient to chase down every `Fetch` and alter it. The second approach, adding a property to the derived file manager, permits the fetch algorithm to be specified once at the beginning of the program.

```
MyFileManager CLASS(FileManager)
FetchOnlyType  BYTE
Fetch          PROCEDURE, BYTE, PROC
                END
```

The compiler can distinguish between the new `Fetch` method and the one specified in the standard ABC `FileManager` class because their prototypes can be differentiated by the rules for procedure overloading. (See pp. 89-90 in the Language Reference Manual.)

### Building A Skeleton

A residence for record fetching logic having been established, the logic itself can be added.

First, set up an `EQUATE` for each value `MyFileManager.FetchOnlyType` can assume:

```
                ITEMIZE, PRE(FetchMethod)
GetByPosition EQUATE
GetNext       EQUATE
                END
```

Although these are the only two options at the moment, setting up this structure provides a clean way to add other options in the future.

The new `Fetch` method, by using the `EQUATEs`, becomes self-documenting.

```
MyFileManager.Fetch PROCEDURE
ReturnValue BYTE, AUTO
CODE
ASSERT( SELF.FetchOnlyType )
SELF.Open
SELF.UseFile
CASE SELF.FetchOnlyType
OF FetchMethod:GetByPosition
! "get by position" code here ...
OF FetchMethod:GetNext
! "get by set/next" code here ...
END
IF ReturnValue
```



```

CLEAR(SELF.File)
ReturnValue = SELF.Insert()
END
SELF.Close
RETURN ReturnValue

```

This "shell" contains everything except the actual logic to access the configuration record if it exists. In adapting the logic from [Steve's article](#), I've made a few assumptions based on my own expected use of the method:

- I may want to access the configuration record in a context other than an update form, so I've removed logic that refers to "SetupForm" from Steve's code.
- I should never call this method unless I've set the FetchOnlyType property. The ASSERT protects against this. Since this would be a programming error as opposed to a data-related condition, I don't need user-friendly error messages for this situation.
- If I ask for a record and can't find one, I always want to add the record.

Your programming style may suggest a different set of design assumptions.

### Putting Meat On The Bones

The only thing left to do is add appropriate code for accessing the configuration record. Whatever the method used, ReturnValue should be set to zero if the fetch is successful, something else if not. The CASE structure is the only part of the shell which changes:

```

CASE SELF.FetchOnlyType
  OF FetchMethod:GetByPosition
    GET(SELF.File,1)
    IF ERRORCODE()
      ReturnValue = Level:Notify
    ELSE
      ReturnValue = Level:Benign
    END
  OF FetchMethod:GetNext
    SET(SELF.File)
    ReturnValue = SELF.Next()
END

```

The new method behaves very much like the standard ABC Fetch except that it doesn't require specification of a key, it expects one and only one record in the file, and, if the file is empty, it adds a cleared record.

### Adding The New Class To The ABC Library

A little plagiarism is a wonderful thing. All the information needed to make making the new class look to Clarion like an ABC class is sitting in the \LIBSRC directory. First, set up files for the new class prototypes and methods. Call them something like MyClasses.inc and MyClasses.clw. Referring to similar files shipped with Clarion, set up these two files to match their style.

Here's the general setup of MyClasses.inc:

```
!ABCIncludeFile
  OMIT(' __EndOfInclude__ ', _MyClassesPresent_)
  _MyClassesPresent_ EQUATE(1)
    INCLUDE('ABFILE.INC'), ONCE
  ! class prototypes here ...
  __EndOfInclude__
```

The comment (!ABCIncludeFile) tells the IDE that this code follows the ABC pattern and should be treated as any other ABC include file. The OMIT structure lets this file be included anywhere the definitions are needed without fear of duplicating those definitions. For example, the definitions of the ABC FileManager are included above so that they can be used in the definition of the MyFileManager class.

The new ONCE attribute provides another mechanism for avoiding duplicate definitions, but that protection depends on the ONCE attribute appearing in every INCLUDE. Older code may still require the protection provided by the OMIT structure, so it's a good idea to leave it in place.

A little more creative plagiarism provides the general setup of MyClasses.clw:

```
MEMBER
  _ABCDllMode_ EQUATE(0)
  _ABCLinkMode_ EQUATE(1)
  MAP
  END
  INCLUDE('MyClasses.inc')
```

The MEMBER statement identifies this as a source module, something the compiler needs to know. The MAP structure is required, since it causes the compiler to include prototypes for Clarion language statements and functions. The INCLUDE of MyClasses.inc makes the new class definitions available to code in this module and, if the nest is not too deep (LRM page 95), also includes definitions from ABFILE.INC.

The two equates, for \_ABCDllMode\_ and \_ABCLinkMode\_, implement the ABC library's method of determining where and how these methods will be linked and referenced. These definitions work with attributes in each CLASS statement which are required and will be added next.

In the skeleton version of the CLASS prototype, some necessary attributes were ignored. The full statement should have been:

```
MyFileManager CLASS(FileManager), |
    TYPE , |
    MODULE('MyClasses.clw'), |
    LINK('MyClasses.clw', _ABCLinkMode_), |
    DLL(_ABCDllMode)
```



The `TYPE` attribute identifies this class as a prototype only. Actual instances of the class will be created (instantiated) when needed. The `MODULE` attribute tells the compiler where to find the code for the class methods. The `LINK` attribute tells the compiler to compile and link these methods if, and only if, `_ABCLinkMode_` is True. The `DLL` attribute tells the compiler to look for these methods in another DLL if, and only if, `_ABCDllMode_` is True.

The full code for `MyClasses.inc` and `MyClasses.clw` is available for [download](#) at the end of this article.

## Have Hammer, Need Nail

Having built a new class, the next challenge is to use it. There are at least two likely situations:

- Access within the context of some other process
- Access in conjunction with an update form

The first type of use is easy. Just use the `Fetch` method as you would the standard ABC `Fetch`:

```
IF NOT Access:MyConfigFile.Fetch
  ! do something, possibly including ...
  Access:MyConfigFile.Update
END
```

(This code assumes that nothing goes wrong during the update. You are of course free to be as conservative as the situation warrants.)

The second is even easier. In the `ThisWindow.Init` method of a form, before the code stores `GlobalRequest`, insert:

```
IF Access:MyConfigFile.Fetch
  RETURN Level:Fatal
ELSE
  GlobalRequest = UpdateRecord
END
```

This allows the form to be called directly from a menu and, no matter how it is called, causes it to update the one and only record in our configuration file.

To make this capability available for a particular file, set `Access:MyConfigFile.FetchOnlyType` to a value indicating how the single record should be accessed. This can be done anytime after the `FileManager` instances are initialized and before the `Fetch` is called.

## Building A Wrapper

It would be nice if all of the necessary housekeeping could be wrapped up in a simple package so that implementing a configuration file would require nothing more than, say, adding a word to the user options of that file in the dictionary. Making things that simple is probably overkill, though, since most projects will have only one file of this type.

An application level extension template can do the same thing cleanly and without the extra processing needed to check every file in the dictionary for a user option. Given the name of a file, the template can initialize the corresponding file manager's `FetchOnlyType` property and add setup code to any form procedure which uses the file.

This is as good a time as any to set up a file for home grown templates. Call it `MyTemps.tpl`. A single line identifies the template chain:

```
#TEMPLATE(MyTemps,'Homegrown Templates'),FAMILY('ABC')
```

Another line establishes the extension template:

```
#EXTENSION(ConfigFile,'Implement Configuration File'),APPLICATION,MULTI
```

The `APPLICATION` attribute tells the generator that this extension is applied at the application level rather than the procedure level. The `MULTI` attribute permits more than one instance of the extension in a single application.

A couple of lines of documentation describing what this template is supposed to do are in order:

```
#DISPLAY('This extension adds code to handle a specified file')
#DISPLAY('as a configuration file containing one and only one')
#DISPLAY('record. Forms for which this file is the primary file')
#DISPLAY('can be called directly without an intervening browse.')
#DISPLAY(' ')
```

A prompt allows specification of the file:

```
#PROMPT('Configuration File:',FILE),%MyConfigFile,REQ
```

There are at least two ways to specify the access method. The easiest is to add a second prompt (or more accurately, prompt structure):

```
#PROMPT('Choose Access Method:',OPTION),%MyAccessMethod
#PROMPT('Get by position',RADIO)
#PROMPT('Get next',RADIO)
```

Another alternative is to have the template select the access method based on driver. This has the advantage of hiding the access method from the programmer, thereby reducing the number of things that have to be remembered when using the template. A good place to put this selection logic is in an `#ATSTART` section:

```
#ATSTART
  #FIX(%File,%MyConfigFile)
  #CASE(UPPER(%FileDriver))
  #OF('CLARION')
  #OROF('DBASE3')
  #OROF('DBASE4')
    #SET(%MyAccessMethod,'Get by position')
  #OF('TOPSPEED')
    #SET(%MyAccessMethod,'Get next')
  #ELSE
    #ERROR('File driver not recognized by ConfigFile extension')
  #ENDCASE
#ENDAT
```

At this point everything necessary to generate code is in place. First, provide for initialization of `Access:MyConfigFile.FetchOnlyType`:

```
#AT(%ProgramSetup)
  #CASE(%MyAccessMethod)
  #OF('Get by position')
Access:%MyConfigFile.FetchOnlyType = FetchMethod:GetByPosition
  #OF('Get next')
Access:%MyConfigFile.FetchOnlyType = FetchMethod:GetNext
  #ENDCASE
#ENDAT
```

Adding code to the input form is a little trickier. The basic structure is easy:

```
#AT(%WindowManagerMethodCodeSection,'Init','()',BYTE'),PRIORITY(0)
IF Access:MyConfigFile.Fetch
  RETURN Level:Fatal
ELSE
  GlobalRequest = UpdateRecord
END
#ENDAT
```

Specifying `PRIORITY(0)` puts the code at the very beginning of the method, which is necessary because `GlobalRequest` is internalized very early in the initialization process.

This code should only be generated if the window is an update form for `MyConfigFile`. The template language provides a `WHERE` attribute to allow this kind of selection. With that attribute in place, the `#AT` statement becomes:

```
#AT(%WindowManagerMethodCodeSection,'Init','()',BYTE'),      % |
                                PRIORITY(0),                  % |
                                WHERE(%ProcedureCategory = 'Form' % |
                                    AND %Primary = %MyConfigFile)
```

One of the benefits of writing articles is that you learn things in the process. Until I had to split a template instruction to fit the printed page I was unaware that the template language has a line continuation symbol and that it differs slightly from the continuation symbol used in Clarion code. You'll find it documented on page 503 of the Programmer's Guide.

Unfortunately, when I tried to use this syntax in an example, the registry rejected the template. You'll find a note which suggests that this might happen on page 525 of the Programmer's Guide. So the continuation symbols (`%|`) in the above example are there for readability only and should not be used in actual code.

The point here is not that the documentation is bad. In fact, it's very good. But the template language has always had more little vagaries and nuances than the Clarion language itself, which makes documentation a daunting task. With both template and Clarion code, I read the documentation, code accordingly, test, and modify until it works. But with templates I don't worry too much if I can't explain in detail why what works, works.

Testing also reveals that `%Primary`, a built-in symbol which the documentation suggests should contain the label of the procedure's primary file, is blank when the `WHERE` clause is evaluated. Why? I don't know. But replacing `%Primary` with `%File`, a multi-valued built-in symbol, works. Why? I don't know.

## Applying The Tool

The three files which define the MyFileManager class and the ConfigFile extension are included in a [downloadable ZIP file](#). Using them is very simple:

- Place MyClasses.inc and MyClasses.clw in your Clarion LIBSRC subdirectory.
- Place MyTemps.tpl in your Clarion TEMPLATE subdirectory
- Register the ConfigFile extension.
- For any file that you want to handle as a configuration file, add an instance of the extension to your application's GLOBAL properties.

You will also need to tell the generator to use MyFileManager instead of FileManager for any configuration files. You can do this either on the Individual File Overrides tab or the Classes tab. The difference will be whether all files use MyFileManager (Classes tab) or just the ones you want handled as configuration files (Individual File Overrides tab).

I usually choose the more general case, expecting to add other functionality to the derived file manager.

In summary, once you have found a solution to a particular coding problem, it makes sense to take the extra step and build tools which implement that solution. Clarion's tool-building facilities are within the abilities of the average programmer, and skill in using those facilities improves rapidly with practice. I hope this example not only proves useful to you, but tempts you to build other tools on your own. You have nothing to lose and a world of productivity to gain.

## A Correction

After this article first appeared, I received an email from Jan Jacob de Maa. He had downloaded the associated ZIP file and was trying to use it. Unfortunately, he was encountering compile errors.

Jan Jacob's experience led to the discovery of three errors in the article, which errors are repeated in the ZIP file:

- In the header for the MyFileManager class, an underscore is missing in the DLL attribute. It should read `DLL(_ABCDllMode_)` rather than `DLL(_ABCDllMode)`. The missing underscore causes the DLL attribute to remain off when it should be on, wreaking havoc when compiling and running in 32-bit mode.
- In the wrapper template, the line `IF Access:MyConfigFile.Fetch` is missing the parentheses necessary to tell the compiler that this is a function rather than a variable. It should read `IF Access:MyConfigFile.Fetch()`.
- Also in the wrapper, the priority of zero which was given for code to be inserted in `ThisWindow.Init` causes the wrapper to execute its `Fetch` operation before the file is open. Changing the priority to 8000 places the code correctly. Also, because at this new position `GlobalRequest` has already been internalized, `GlobalRequest = ChangeRecord` has been changed to `SELF.Request = ChangeRecord`.

[Download the updated source code](#)

---

[Nik Johnson](#) stumbled into the programming racket in 1959 when his boss at Grumman Aircraft insisted on his attending a Fortran class. Since 1986 he has been using Clarion to help clients solve information handling problems.

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).

In This Issue

[Tom Hebenstreit  
New Reviews  
Editor](#)

Posted on August  
10, 1999

[Working With  
Control Files II](#)

Posted on August  
10, 1999

[Is It Six Months  
Already?](#)

Posted on August  
10, 1999

[Updated COSP](#)

Posted on August  
10, 1999

[New Open Source  
Products](#)

Posted on August  
10, 1999



# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## Press Release

### We're Six Months Old!

Okay, so six months isn't old. But I can hardly believe that it's already been half a year since Clarion Magazine first published. Over that six months Clarion Magazine has published a wide variety of articles, reviews, interviews, and news items. Here's a sampler:

Several article series have been popular with readers. The Novice's Corner articles have covered the basics of the [Clarion development environment](#), [understanding templates and embeds](#), [designing databases](#), and [many-to-many relationships](#). David Bayliss has written extensively on the internals of the ABC class library, to date covering [ErrorClass](#), [FieldClass](#), [ConstantClass](#), FileManager (in parts [one](#), [two](#) and [three](#)). David continues this series with the RelationManager. He has also written on [Wizatrons](#) and has done some interesting analysis of [Clarion Challenge Results](#).

The ABCs of OOP series by Dave Harms (in parts [one](#), [two](#), and [three](#) to date) covers object-oriented programming principles and practices in the context of ABC. Steve Parker is a regular contributor to Clarion Magazine, and his works include a discussion of [NAME\(\)](#) and a [three part series](#) on handling sort orders.

One regular feature is the Clarion Advisor, which offers advice on common programming problems. The Advisor has covered topics such as [debugging using project files](#), [Topspeed driver error codes](#), [custom editor colors](#), [fast ASCII file access](#), and [more](#).

Clarion Magazine frequently interviews movers and shakers in the Clarion software development world. [Roy Rafalco](#) spoke with Clarion Magazine shortly after his appointment as [Topspeed's](#) new CEO, and offered many insights into how Topseed functions and where the company is headed. You can also read about [Ragnar Hellspong](#), creator of [ForKeeps](#), and the [three Clarion developers](#) behind the internet's Clarion knowledgebases.

We're also fortunate to have as regular contributors Andy "Cowboy" Stapleton, who rides herd on a variety of SQL databases, and Larry Teames, one of the best-known third party vendors and an expert on reporting. You can find their articles, and a whole lot of other things, by going to the [Clarion Magazine search engine](#). Search for, respectively, "Cowboy" and "Teames".

**BKO**  
Enterprises, Inc.

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

etc  
2000

If you're interested  
take our poll &  
let us know!

Clarion Magazine also contains product reviews (such as Datamatrix's [Xplore templates](#) and Sterling Data's [Searchflash](#)) and the ever popular [Clarion Challenge](#) series. Tom Hebenstreit has joined Clarion Magazine (from the now defunct ClarionOnline) and will be heading up product reviews.

For the truly hard core hand coder there are in-depth treatments like Jim Kane's assembler-driven technique for [calling OLE methods](#). And for those who like to download and read at their leisure (leisure, what's that?), each issue of Clarion Magazine is also available in [PDF format](#). PDFs are also recommended if you want to print an issue.

This is a sampling of what's been available in Clarion Magazine over the past six months. For a complete listing please see the [site index](#).

### What's Next?

We're heading into our second half-year with great anticipation. September brings the Clarionfest in Florida, [DevCon '99](#). If you can't make it to Fort Lauderdale this year, and even if you can, tune in to Clarion Magazine for daily coverage of the conference proceedings in word, pictures, and maybe even some MPEG video.

We have a lot of terrific articles waiting in the wings as well. Over the next two months Jim Kane concludes his two-part series on OLE, Bruce Gilham serves up rules for successful application development, and Steve Parker and David Bayliss (aka DAB) trade ideas on file aliases. DAB also continues with his series of articles on the inner workings of ABC, and Larry Teames reveals more reporting secrets. This fall also sees the return of Andrew's Kitchen!

If this sounds like a packed publishing schedule, it's because it is, but there's always room for one more article. Contributions to Clarion Magazine are welcome. If there's an article you'd like to write, send an email with a brief outline to [editor@clarionmag.com](mailto:editor@clarionmag.com).

Dave Harms  
Publisher

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).

In This Issue

[Tom Hebenstreit  
New Reviews  
Editor](#)

Posted on August  
10, 1999

[Working With  
Control Files II](#)

Posted on August  
10, 1999

[Is It Six Months  
Already?](#)

Posted on August  
10, 1999

[Clarion News -  
August 1999](#)

Posted on August  
10, 1999

[Updated COSP](#)

Posted on August  
10, 1999

[New Open Source  
Products](#)

Posted on August  
10, 1999





# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## The Clarion Open Source Project

### Product Index

Updated September 05, 1999

The following products are available from Clarion Magazine under the [Developers Open Source Public License \(DOSPL\)](#).

- [Debugging/Profiling](#) - These classes from CoveComm Inc. generate trace logs and application execution logs. Template included. Version 1.1
- [BitList Management](#) - Class and template from JS Software (Jeff Slarve) to bitslice LONGs and display checkboxes for setting the individual bits. Version 0.9912. Very slick.
- [File Utilities](#) - A set of classes by Patrick O'Brien to handle file dialogs and splitting file names. Includes an example application.
- [Debug Classes](#) - Richard Rogers' class and template for debugging/tracing.
- [Graphing Classes](#) - Chris Behling's example application does graphing using all-Clarion code.
- [Trackbar/Slider](#) - Pierre Tremblay's 32-bit slider class and template as described in his Clarion Magazine [article](#).
- **NEW!** [Graphing Learning Example](#) - James Cooke's learning example does OOP graphing using all embed points.

If you've placed some code under the DOSPL and you'd like to have it listed here, email [cosp@clarionmag.com](mailto:cosp@clarionmag.com)

**BKO**  
Enterprises, Inc.

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

**etc**  
2000

If you're interested  
take our poll &  
let us know!

In This Issue

[The Other Way To Use OLE - Part 2](#)

Posted on  
September 7, 1999

[The Cranky Programmer - Install THIS!](#)

Posted on  
September 7, 1999

[September News](#)

Posted on  
September 7, 1999

[Open Source Update](#)

Posted on  
September 7, 1999

[The Clarion Magazine Technology Poll](#)

Posted on  
September 7, 1999

[Correction: Class And Wrapper For Handling Control](#)



[Files](#)

Posted on  
September 7, 1999

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).



# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## Product Review

### NextAge Imaging Templates v1.06

Reviewed by Tom Hebenstreit

By my count, there are currently 18,169 files in the Windows folders on my machine (give or take a few temp files). Now, you just know that there have to be some useful things lurking around in there, don't you?

Well, there are (really!), and the templates that I'll be looking at in this review help bring to light one of the more useful (and little known) tools that all Windows 9x and NT machines come with: a full set of scanning and imaging tools. They are provided as a set of OCX controls (ActiveX in modern parlance), and you'll sometimes hear them referred to as the Wang imaging controls. (Wang originally developed and licensed them to Microsoft, but now Eastman Software develops them.)

The NextAge Imaging Templates, created by [NextAge Consulting](#), are a set of templates designed to wrap around and simplify the process of using that built-in imaging subsystem. The templates are designed for Clarion5, using either ABC or legacy template chains.

#### Major Features

Between the templates and the underlying image controls, highlights include:

- Support for scanning (black and white, color, photos, any resolution – basically, whatever your scanner supports)
- Pages within a scanned document can be viewed one at a time (with zooming, rotating and scaling) or as a series of thumbnails (miniature versions of the scanned images)
- Printing scanned images
- Multiple methods of storing, managing and retrieving scanned items
- No handcode required to implement any of the above

As a bonus, the package includes an additional set of free non-image-related templates.

#### Installation

I downloaded the templates from the NextAge web site as a single file self-installing package. The opening WISE installer screen stated that it was installing v1.06 of the templates, so I was a bit surprised when the rest of the package referred to version 1.05. In looking in the templates themselves (nicely commented, by the way) it seems that 1.06 was the version that was installed.

Installation proceeded smoothly enough but missed what I consider to be some of the basic features that separate the good installs from the merely functional ones. For example, it didn't automatically detect the location of my Clarion 5 folders, nor did it display the documentation when finished.

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

If you're interested  
take our poll &  
let us know!

Bottom line: The install works but could use a bit of polish to make it smarter and friendlier.

## Implementation

Before adding the NextAge Imaging templates to an application, you first need to decide how your program is going to handle storing the scanned images.

Note: All scanned images are created and used in the TIFF graphics format, regardless of how you choose to store them. Also, a TIFF file can contain more than one image; so no matter how many pages you scan for a particular document, you will still end up with a single disk file.

Basically, you have two choices: store the TIFF images as individual files on disk or store them inside your data files as a BLOB (Binary Large Object). Each method has its pros and cons:

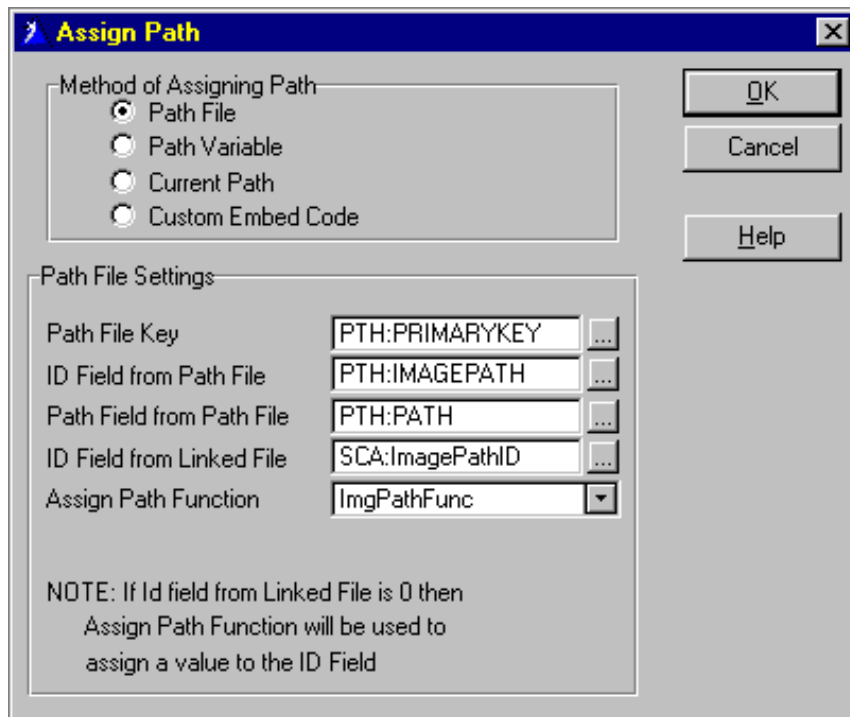
- Using a BLOB, everything is conveniently in one place: you have one data file and it contains both the image and any associated information that you want to store with it. Having everything in one file is also the primary weakness of this method; if the file is damaged, you could lose everything. If you have high volume and/or large images, you could also see file performance suffer as the file gets larger (and larger and larger). The final point to keep in mind is that not all file systems and drivers even support the use of BLOBs.
- When storing scans as individual files, all you need in your data files is some kind of pointer to the name (and possibly the location) of the associated scan file. The advantages of this method are many: your data files remain small (and thus fast to search), data file corruption does not lose image data, and you can use a wider variety of data formats and drivers. The downside is that you have to manage potentially thousands (or hundreds of thousands) of files. Fortunately NextAge provides some built-in methods for automatically naming and filing the scan files. Even better, you can specify a maximum size for scan folders. When that size is exceeded, your application will automatically create a new folder and start putting new images there. This can make it easy to archive older images to, for example, a CD-ROM. Just update the storage path for the folder in one record of an image path file and bingo – your thousands of data records now know where to find their associated images. Very slick.

OK, enough background. To test NextAge's claim that no hand code is required, I created a sample Clarion 5 ABC Wizatron application and then followed the instructions in the docs to add imaging. I decided to use the non-BLOB storage method and the built-in procedures for naming and managing the scan files as well.

Everything went well until I was faced with filling out the prompts to specify how the various data and image path files would be connected (see figure 1). The docs are a bit fuzzy at this point, so it was here that I found the NextAge demo app to be invaluable. By basically duplicating what was done there, I finished the process in next to no time.

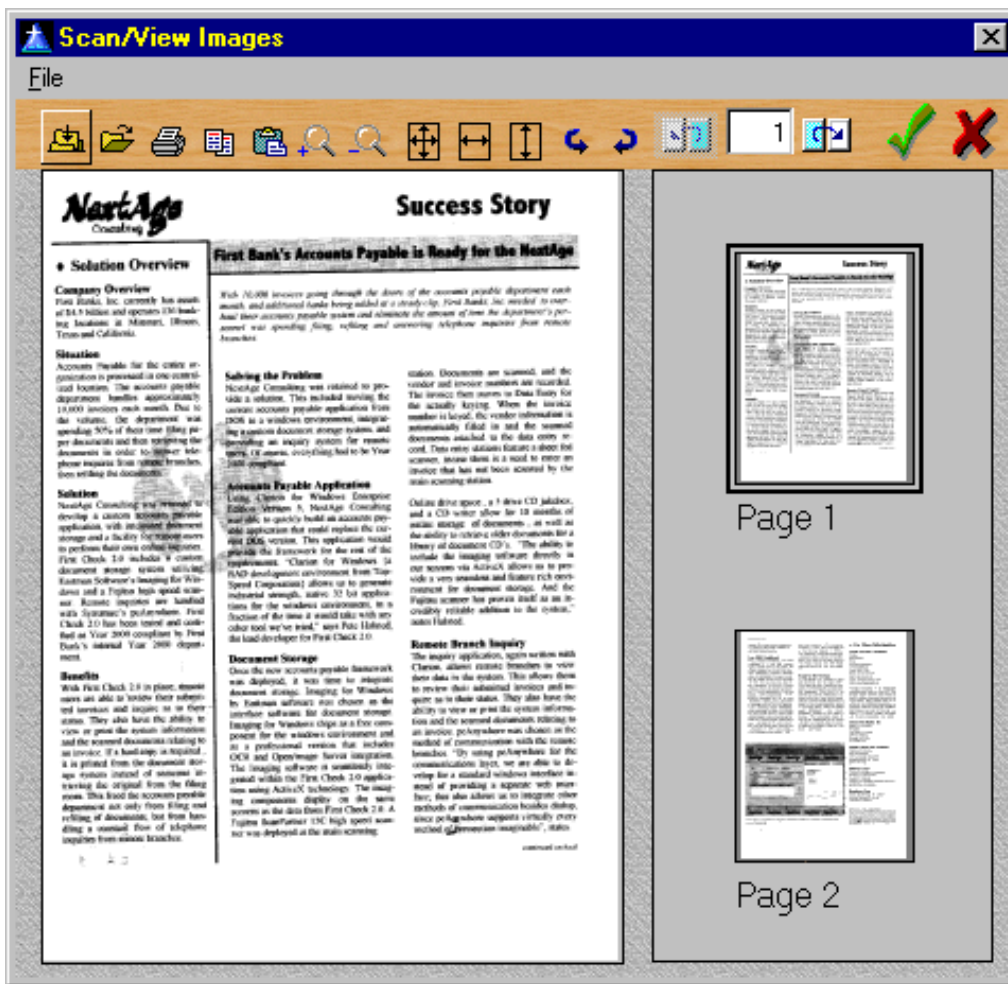
Figure 1. Template prompts for linking images, file paths and data records

Figure 1. Template prompts for linking images, file paths, and data records.



A quick, error-free compile and I could add records, scan and attach multi-page documents, view images and print them out. NextAge was right – nothing but templates. If you need to do something special, there are embed points where you can get down and dirty within the templates as well.

Figure 2. The default NextAge Image control procedure.



## Room For Improvement

There are a few minor places where the templates could be improved:

- I'd like to see more of the internal imaging OCX properties exposed on the templates. To illustrate, the word "Page" is hardcoded in the templates as the caption to be displayed under thumbnails. If, for example, I was scanning engineering documents, I might want the caption to be "Sheet" instead.
- The Maximum directory size prompt doesn't say what the unit is (bytes, K, megs). It turned out to be bytes. I'd like to see it on the prompt itself to save a trip to the docs.
- As shown above in figure 2, the main Imaging procedure template is pre-formatted to have the same look as the demo. Ideally it should be more generic so that you don't have to strip out the toolbar and window background images every time you use the template.

All in all, though, the templates were very easy to use.

## Performance

What can I say? It worked. Performance was really more dependent upon the underlying subsystems such as the scanner, drivers and CPU speed than on anything the templates do. I encountered no problems at all with either the generated code or the OCXs.

## Documentation

Documentation is provided in the form of an Adobe Acrobat PDF file, and thus requires that the free Acrobat reader be installed on your machine.

Totaling 13 pages, the docs are nicely laid out and contain some very useful background information on the imaging OCXs, image storage

formats, file sizes and more. A template reference section covers each of the templates and explains each prompt on them. The PDF format makes it a snap to print, which is a good thing because there is no online Windows help file for the templates

My only suggestion would be to add a bit more detail about what was installed and where it went. For example, the documentation mentions that a demo application was installed, but doesn't tell you what the file is actually called. In point of fact, two demo apps are installed, one using ABC and one using the legacy (procedural) templates. Additionally, there is no mention of how to register the templates and what the names of the two template chain files are. This leaves you guessing as to which files need to be registered.

As I mentioned above, the demo apps are also a good guide to the various ways the templates can be implemented.

For more adventurous souls, the documentation also points you to the Eastman Software web site where you can download free documentation for the imaging controls themselves. Using these, you can examine the source code generated by the templates and get a complete understanding of what is happening behind the scenes.

## Technical Support

NextAge lists their address, phone, fax, web site and email address as ways of contacting them, so you have no shortage of support options. I chose to use email, and all questions were answered in a comprehensive manner the same day that I sent them. Even better, my suggestions for improvements were met with enthusiasm, and added to the list for future releases. NextAge also monitors the TopSpeed third-party newsgroup, and a question I posted there was answered within 24 hours.

If they keep it up, support like this will land them in the top tier of Clarion third-party vendors.

## A Little Something Extra

As a bonus, the NextAge Imaging Templates package also contains their set of free Clarion5 ABC templates. Functions include management of application icon and background images, simple application security, combining multiple types of lookups into a single file and a handy template for providing the "build" date of the application (useful for keeping "about" and splash screens up to date).

Like the imaging templates, documentation is provided as a PDF file. Unlike the imaging template docs, the free templates have a very good section on what was installed and how to register the templates (hint, hint).

Note: These free templates can also be downloaded directly from the [NextAge website](#) – no purchase necessary.

## Summary

I found the NextAge Imaging templates to be a good example of someone adding value to another product by simplifying and enhancing its use. It not only makes imaging easy to add to your applications, but provides some very useful methods of managing all the files or data that your scanning applications will invariably collect.

Do you absolutely need them to use the Windows built-in imaging sub-system? No, you don't, but I guarantee that it would take you a lot more time and money than these templates cost to figure it all out and get it working all by yourself.

Bottom line: If you need scanner/imaging support in your applications, I highly recommend these templates.

## In This Issue

[Product Review: Imaging Templates](#)

Posted on August 17, 1999

[Interview: College Aid Calculator](#)

Posted on August 17, 1999

[The SQL Answer Cowboy](#)


Posted on August 17, 1999





[Undocumented Debugging](#)

Posted on August 17, 1999

[Clarion News - August 1999](#)

Posted on August 17, 1999

PRODUCT RATING	
Overall	
Ability to do the task	Very Good
Ease of use	Very Good
Ease of installation	Good
Documentation	Good
Technical support	Excellent
Black box DLLs/LIBs	No*

LEGEND	
First class all the way	
More than adequate	
Barely adequate	
Don't even think about it	

\* While the templates are purely Clarion source, the underlying imaging OCXs are, of course, a closed box.

The NextAge Imaging Templates list for US\$149, and are available directly from [NextAge](#) or through the [TopSpeed Accessories program](#). For more information on the templates, ordering them, links to Eastman Software (for detailed OCX info) or to download the free NextAge Imaging demo, visit their web site at <http://www.thenextage.com>

Vendor Comments from Pete Halsted of NextAge Consulting:

We are committed to continuing to enhance the imaging templates, and all points and suggestions that Tom raised during his review will be seriously considered. Many of the enhancements in the last few releases came directly from suggestions from our customers. As to the issue of exposing more of the underlying OCXs we continue to do this, based on customer suggestions. However, since we are working with OCXs there are times that we will run into Clarion/OCX issues. When the new Clarion OLE interface is released we expect most of these issues to no longer exist. There will be a "professional" version of the imaging templates that supports annotations and OCR. The professional version will require the Imaging Professional controls from Eastman Kodak (not free), which are a superset of the Free controls. An official price or release date has not been set, but I'm sure the good folks at Clarion Magazine will let you know when they are available.

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).





# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## Feature Article

### Interview: College Aid Calculator Developer Steve Brown

Although Clarion as a development tool is little known, some of the products created with Clarion have gained wide acceptance. One of these is the College Aid Calculator, available on the Internet at <http://www.collegeaidcalculator.com>. Clarion Magazine recently interviewed Think Ahead developer Steve Brown.

Clarion Magazine: Your product, the College Aid Calculator, is one of the most widely-distributed Clarion programs ever created. How many copies are currently in use?

Steve Brown: Actually, that is a difficult number to come up with. We sold around 250,000 copies of the software last year, but almost everyone who uses it gets it for free. We sell mostly to lending organizations (Fleet, Bank America), as well as to colleges and universities. The program is customized with their logos, loan information, etc., and then given to prospective clients and students as a marketing tool.

How do you go about customizing the application for a particular client?

Almost all of the customization info (logos, help messages, menu options, etc) are maintained in configuration files, and can be changed without recompiling the program. Some customers have even changed the name of program to make it look like their product.

What does the College Aid Calculator do?

In a nutshell, the program figures out how much parents and students are expected to pay towards their college education, and how much of college costs will be paid by scholarships, grants, and loans. It also explains the financial aid process, allows parents to try different saving and loan scenarios, and prints reports to help students fill out financial aid forms.

What features of Clarion made it a good choice for this application?

Size, for one. The program, the Clarion runtime libraries, the help files, and the setup program all fit on one diskette. The pricing and distribution method require that it is all stuffed on one 3 1/2" disk, and not a CD. Even with the space crunch, we are able to give field-by-field help and include a book on college aid in .hlp format on the disk.

The other big factor was 16-bit development. Although we will probably be moving to 32-bit by the end of the year, we needed to support Windows 3.1 when we started. Clarion allowed us to create a product that had the look of a Windows 95 program but would run on both platforms.

Which version of Clarion are you using?

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

If you're interested  
take our poll &  
let us know!

The program is completely written in C5EE. The previous version was written in CW 2.003, and needed to be rewritten to make customizing the product an external process. C5 was just going into beta when I started the project, so I skipped over C4 and went right to C5. Fortunately, our project and C5's beta cycle finished at about the same time.

Did you use any third party tools in creating the College Aid Calculator?

Yes. We use File Manager 2 from [CapeSoft](#), and are extremely pleased with it. We also use [TopSpeed's](#) Business Math Library, and the Internet Framework Templates from [LogiCentral](#) in our web version.

How is your development team structured?

All of our developers and testers are in different locations, and work remotely via the Internet. The core development team gets together about every other month, but everything else is done using phone, fax, and email.

What was your biggest challenge in creating the application?

Coming up with the user interface took a lot of work. College aid is a complicated subject, and we wanted to make the program easy to use. Also, we are a small company and couldn't afford a huge customer support staff. We have spent a lot of time and effort developing the field-by-field help and accompanying guide to college aid. The software is distributed without any paper documentation, yet we receive very few support calls.

Marketing is often the biggest hurdle facing vertical market developers. What is the key to your success?

Well, picking the right market has been one factor. We knew that most people would be faced with the college finance problem sooner or later, but we found it difficult and expensive to get into the retail market. By customizing our software so it can be used as a marketing tool, we are able get our product out without the expense of retail packaging. FAME (the Finance Authority of Maine), for example, purchased enough copies to hand out to every high school junior in the state of Maine. There is no way we could get that kind of exposure if we had focused only on the retail market.

What tools did you use to create the online help?

We use RoboHelp for the .hlp files distributed with the program. The field-by-field help is just text stored in a .tps file, and then linked to specific sections of the help files when appropriate.

Are you using Wise for Clarion for your installs, or the full version?

We use Wise Install Builder for our installs. We've evaluated most of the different installation packages, but found that Wise had the smallest footprint. We need to heavily customize the installation look and feel for some customers, so we had to upgrade from the Clarion version of Wise.

You mentioned an Internet version of College Aid Calculator.

We do have an Internet version of the program (HTML, JavaScript, and CGI) that uses a Clarion app to do all the calculations. Our formulas are pretty complex, and we wanted both the web and disk versions to work off the same code. So we used the [Internet Framework Templates](#) to write a CW app that receives data from the web, calculates the answer, and then sends it back. This is in the last stage of development, but should be running on several large sites in the next week or two.

Did you consider doing the project in CWIC?

I really wanted to do this project in CWIC, and tried about every way possible to make it work. However, our web version also has field-by-field help, and even the Java-free version of CWIC was just too slow. One of the banks deploying our web version in the next couple of weeks gets about a million visitors a month on their site, and I didn't think CWIC could handle that type of a load.

Is this is because of the overhead of replicating a help window rather than just serving up standard HTML?

Sort of. Each screen has multiple fields, and each field has different help text. The Java version would replicate the help window from the Clarion program. In the Java-free version, I needed to manually add an ONFOCUS event for each field and maintain and display the help outside of the Clarion program in a different frame. But even with Java-free, displaying the 10 or 12 screens with a few fields each was slow. The newer version uses cascading style sheets so all the screens load at the beginning, and the user moves from page to page instantaneously.

How do you handle security and pricing for the Internet version? Or can anyone use that product?

When someone purchases the web version, they deploy it on their site for anyone to use. The pages are customized for that particular customer, so it is unlikely that a competitor would link to the same site for a "free ride". Users of the software aren't asked to enter any identifying information, and none of the data is saved after they leave, so security really isn't an issue.

For more information on [College Aid Calculator](#) contact Steve Brown at [sbrown@linc-net.net](mailto:sbrown@linc-net.net).

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).

In This Issue

[Product Review: Imaging Templates](#)

Posted on August 17, 1999

[Interview: College Aid Calculator](#)

Posted on August 17, 1999

[The SQL Answer Cowboy](#)

Posted on August 17, 1999

[Undocumented Debugging](#)

Posted on August 17, 1999

[Clarion News - August 1999](#)

Posted on August 17, 1999



# Clarion magazine

[Main Page](#)
[Log In](#)
[Subscribe](#)
[Open Source](#)
[Links](#)
[Mailing Lists](#)
[Advertising](#)
[Submissions](#)
[Contact Us](#)
[Site Index](#)
[ClarionMag FAQ](#)
[Download PDFs](#)
[Search ClarionMag](#)

## The SQL Answer Cowboy

Andy "Cowboy" Stapleton is the acknowledged Clarion SQL guru and a regular presenter at Clarion conferences around the world. His company, [Cowboy Computing Solutions](#), produces SQL templates and classes for Clarion.

[Click here](#) to submit your SQL question to Andy.

Jim Kane: Contrary to your [recent] column MSSQL 7.0 does run on Windows 95. I have it installed on my notebook. I can connect other Windows 95 or NT clients to my notebook and frequently do for load testing.

Cowboy: I appreciate your feedback, and you are correct, at least under these conditions: MS-SQL 7.0 will run on Windows 95 if DCOM is loaded or you have the patch that upgrades the Internet Explorer to 4.01. Without those, MS-SQL will not run on Windows 95. In effect you are upgrading to Windows 98 components. I have also loaded MS-SQL 7.0 to my 95 machine by doing the same thing.

Patrick O'Brien: Can a Clarion application using Microsoft Data Engine (MSDE) be run successfully on a Macintosh network using SoftWindows 95? There won't be more than two or three concurrent users. (You didn't say they had to be easy questions, did you?)

Cowboy: The questions you have to answer first are:

1. Does any Clarion program run on SoftWindows 95?
2. Can I load and operate MSDE on the same platform?

What I would do is use a standalone Clarion program and try to run it under SoftWindows. If that runs then you have answered the first question.

Now run MSDE on the Mac and see if you can access the data in Northwind or Public. The last thing on your To Do list is to run a small standalone Clarion program that accesses one of the databases.

Rick Smith: I read your recent SQL comparisons and answers. It seems that you favor Sybase quite a bit. In fact, that's the engine we're planning to use for our upcoming upgrade. However someone was telling me that there may be cause for concern with respect to the financial stability of the company. Can you speak to this issue?

Cowboy: Currently Sybase is up 19% over last quarter. 1998 was a year of concern, but with the restructuring and demand for some of the products, I don't believe it will go away anytime soon. Here is a link that gives you the Wall street press release for Sybase for Q1 99.



If you're interested  
take our poll &  
let us know!

In This Issue

[Product Review:  
Imaging  
Templates](#)

Posted on August  
17, 1999

[Interview:  
College Aid  
Calculator](#)

Posted on August  
17, 1999

[The SQL Answer  
Cowboy](#)

Posted on August  
17, 1999

[Undocumented  
Debugging](#)

Posted on August

17, 1999

[Clarion News -  
August 1999](#)  
Posted on August  
17, 1999

[http://dynamic.sybase.com/press\\_releases/press\\_releases/ExternalItem/0,1099,19700,00.htm](http://dynamic.sybase.com/press_releases/press_releases/ExternalItem/0,1099,19700,00.htm)

[Click here](#) to submit your SQL question to the SQL Cowboy.

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).





# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## The Clarion Advisor

### Undocumented Clarion

by David Harms

The ongoing discussion in the pages of Clarion Magazine regarding debugging prompted one reader to point out the following undocumented debugging functions built into Clarion.

NOTE: As these are undocumented functions and come with no promise of support you should not use them in production applications.

Listing 1 shows a small test program which uses the undocumented debugging functions.

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

Listing 1. An example program which generates an event log.  
program

```

map
  module( 'WSLDEBUG' )
    PrintDebugHex(USHORT),NAME('WslDebug$PrintHex')
    PrintDebugString(STRING), NAME('WslDebug$Print')
    PrintDebugLine(), NAME('WslDebug$PrintLine')
    PrintDebugEvent(), NAME('WslDebug$PrintEvent')
  end
end

Window WINDOW('Debug Test'),AT(,,117,40),|
  FONT('MS Sans Serif',8,,,CHARSET:ANSI),GRAY,DOUBLE
  BUTTON('Close'),AT(36,12,45,14),USE(?Close)
END

code
open(window)
accept
  ! Print the first line of text
  PrintDebugString('Event - Decimal: ' & event() & ', Hex: ')
  PrintDebugHex(event())
  PrintDebugLine()
  ! Print the second line of text

```

```

PrintDebugEvent()
if accepted() = ?Close
    PrintDebugString('closing window')
    PrintDebugLine()
    post(event:closewindow)
end
end
end

```

Running this program simply brings up a small window with a Cancel button. Press the cancel button, and when the program terminates a text file called c5log.txt contains the text in Listing 2.

Listing 2. The event log created by the test program.

```

Event - Decimal: 518, Hex: 206
EVENT:GainFocus          0
Event - Decimal: 515, Hex: 203
EVENT:OpenWindow        0
Event - Decimal: 257, Hex: 101
EVENT:Selected           ?CLOSE          (1)
Event - Decimal: 1, Hex: 1
EVENT:Accepted           ?CLOSE          (1)
closing window
Event - Decimal: 513, Hex: 201
EVENT:CloseWindow       0

```

I've made the log a bit more complicated than it needs to be just to demonstrate the use of the different functions. PrintDebugString just prints whatever string you give it. PrintDebugHex will translate the passed decimal number to a hex equivalent, useful since Windows equates are usually hex numbers. And PrintDebugLine simply prints a CRLF.

As you can see from Listing 1, you can use these functions to display information about the program's events, but a far easier and more complete solution is to just use the PrintDebugEvent function, which prints the Clarion equate of the current event, the field equate of whichever control received the event (a value of 0 refers to the window), and the field equate number.

These undocumented functions make a useful addition to any Clarion developer's toolkit.

[Download the example source file and project.](#)

In This Issue

[Product Review: Imaging Templates](#)

Posted on August 17, 1999

[Interview: College Aid Calculator](#)

Posted on August 17, 1999

[The SQL Answer Cowboy](#)

Posted on August 17, 1999

[Undocumented Debugging](#)

Posted on August 17, 1999

[Clarion News - August 1999](#)

Posted on August 17, 1999

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).





# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## Feature Article

### Alias: Who **Was** That Masked File?

by Steve Parker

"File alias" is one of those concepts that is at once extremely difficult and extremely simple. On the one hand, the documentation is quite sparse. Alias is not part of the language, not a Clarion statement (at least in the context of files - there is an ALIAS statement which changes keycodes, but that's not what I'm referring to). The one reference in the online help is to a FileManager method. Little information and, therefore, guidance, is available. On the other, the notion of referring to something by a different name (which after all is what an alias is) is really quite simple: two names, one object.

The behavior of file aliases is entirely a consequence of some Clarion fundamentals: LABEL, RECORD and NAME. When you understand these concepts, it is almost easy to use aliases to solve real world problems.

#### Why Aliases?

Aliases were introduced to allow multiple relations between files, a feature not supported by the Dictionary Editor.<sup>1</sup> Aliases, therefore, provide a second record buffer on a thread for a file. This second buffer also allows looking up recursively within a single file. Aliases are powerful stuff.

Aliases, therefore, are the answer to the question "How can I be in two different places in one file at one time?" Other ways of saying this include: "How can I do a lookup from a file into itself?" "How can I have two different relations between the same two files?" and "How can I relate one record in a file to other records in the same file?"

Recursive lookups and relations are not easily visualized. Perhaps an example or two will clarify the matter.

A fairly clear example is an employee file in which you want to display the employee's supervisor (a job title file displaying the supervisor's title is much the same thing and can serve as the basis for this sort of lookup). Since a supervisor is also an employee (OK, I know that many oughtn't be), there is an interesting and thoroughly unattractive choice: either you can have two files (two copies of the employee or job title file) or you can have one file. Maintaining only one copy of the file means that you have to look up within the current file.

A recursive lookup like this will not work without a major finagle<sup>2</sup> and that assumes that you can describe it well enough to create a specification (it's not that easy; try it). Two copies of the same file simply for the purpose of a lookup seems...well, stupid, frankly.

In an inventory module, a bill of materials or kit presents a very similar situation. When displaying a BOM or kit, there will typically be a browse of the items comprising the kit. This is a parent-child relationship. The parent inventory item is the kit but the child items are (or were) also inventory items. Even if the component items are stored in a separate file, they start out in the inventory file. When displaying the components,

**BKO**  
Enterprises, Inc.

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

etc  
2000

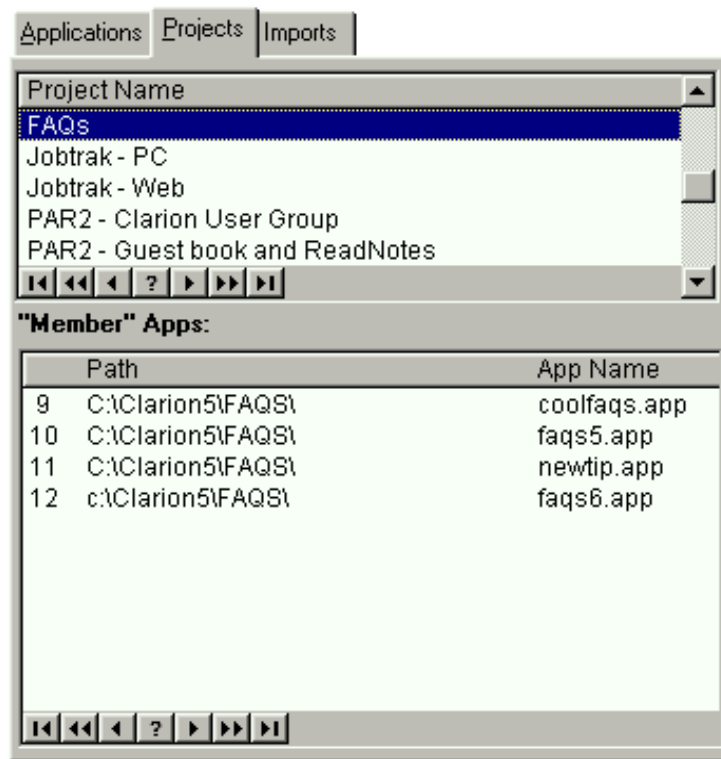
If you're interested  
take our poll &  
let us know!

you need to relate one record in a file to other records in the same file (either directly or indirectly) or keep two copies of the inventory file. When creating the kit, you will need to select items from the inventory file which you are already accessing to insert the kit record. Again, the choices are less than palatable.

My own introduction to aliased files occurred while updating my Go To Lunch batch compiler (available at the [CWICWEB](#) download site).

I needed two different browses of the same file on two tabs. On the first tab is a simple browse of the app list. On the second tab, the app list is displayed filtered (actually, range limited). Sounds like pretty standard stuff, right? On the second tab, however, the file is used as a child of another file (Projects) and the second tab shows a list box from the parent file. The app list is filtered on the currently selected parent record (see Figure 1).

Figure 1. The Go To Lunch Batch Compiler showing an aliased browse ("member" apps).



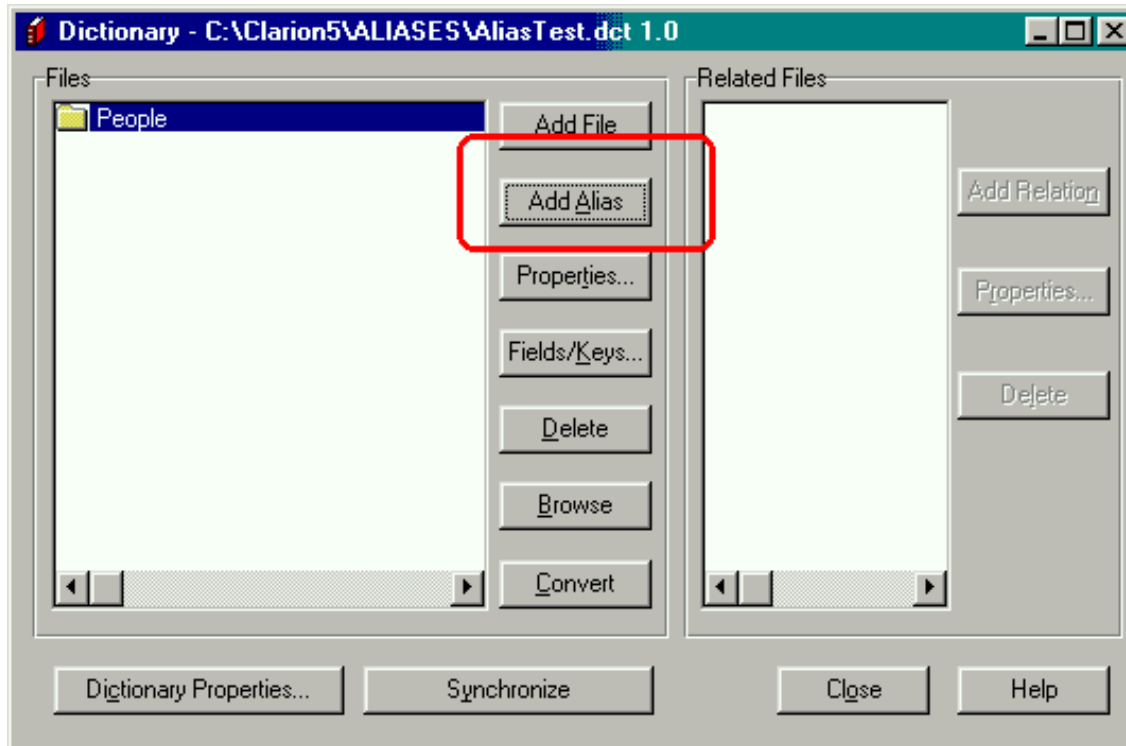
What is not standard is using a single file standalone and as a child in the same procedure. The standalone instance will set the buffer to the last accessed record. The second instance will always range limit the file on the relating key values. Mashed buffers are a certainty: the file will always end up range limited on the linking field value from the second tab. Even though the first browse appears normal, you will only be able to access the record that is active on the link as currently set on the second tab (opened last, the parent file "touched" the target file last and so gets its way).

### Creating An Alias

Before continuing I need to describe how to create a file alias. It's a bit "cart before the horse," but this is the easiest way to present the information.

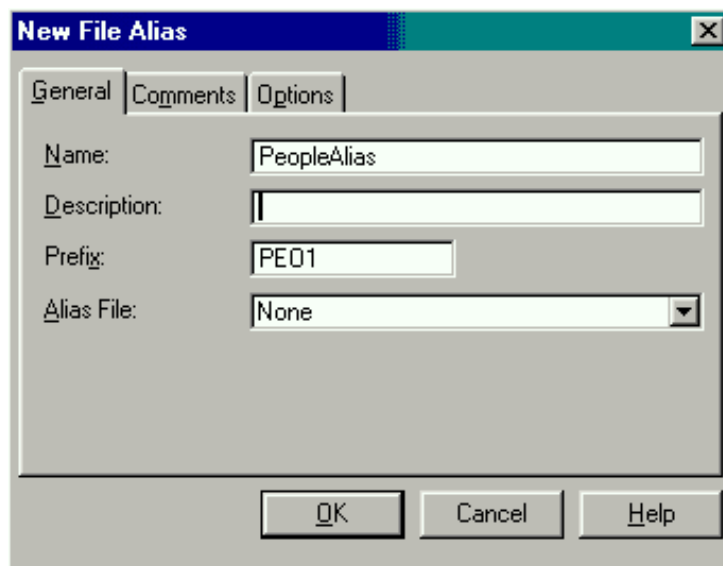
The Dictionary Editor makes creation of aliases simplicity itself: just press the Add Alias button (see Figure 2).

Figure 2. Creating an alias in the dictionary editor.



This will call the File Alias worksheet, shown in Figure 3.

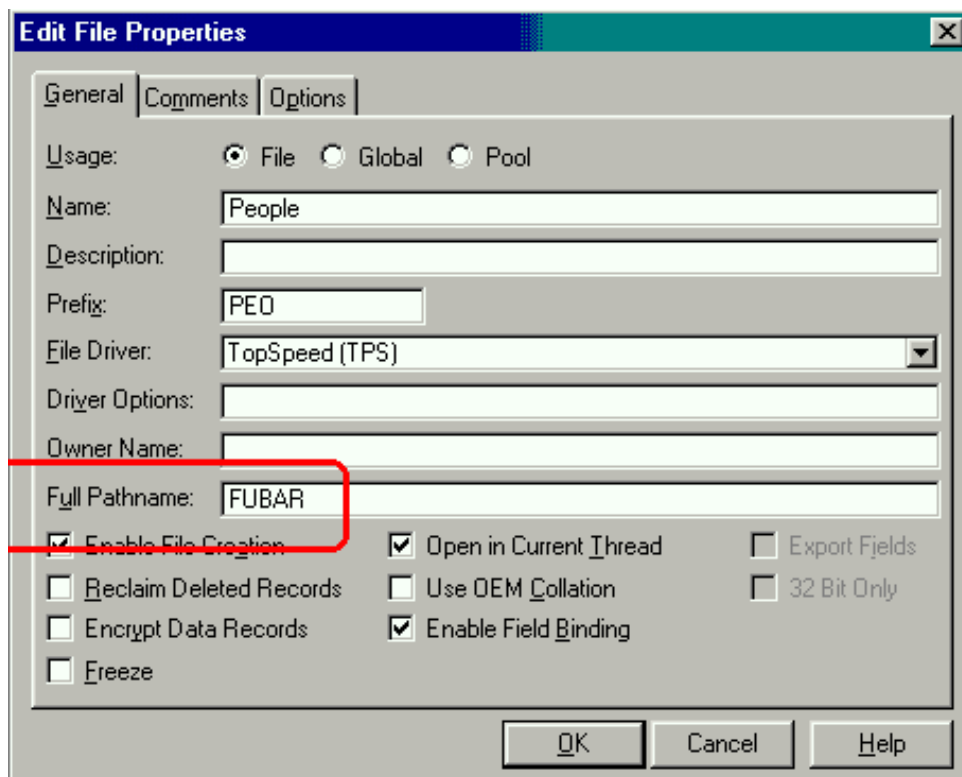
Figure 3. The File Alias worksheet.



If you are not already using the NAME attribute on the file to be aliased, you will be notified that this is required.

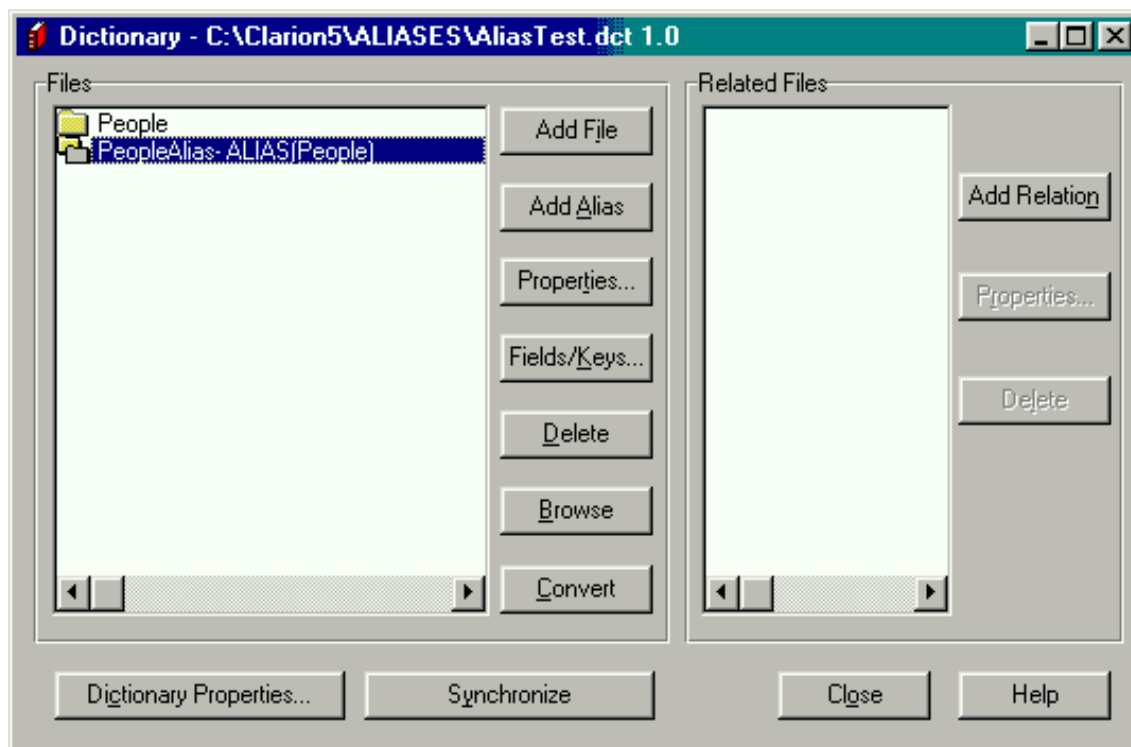
There is no particular reason to use a variable for the NAME, unless you wish to (see [NAME\(\) Comes of Age](#)); you can just specify a DOS file name:

Figure 4. Completing the NAME() attribute for the file being aliased.



When you complete all the prompts, your dictionary will look like Figure 5:

Figure 5. The completed dictionary worksheet.



It is extremely important to note that if you do not complete the `NAME` attribute of the base file, your dictionary will still look like Figure 6. However, when you run the app, a file will be created on disk for the "alias" (if you have file create turned off, you will get an error message). Of course, this defeats the purpose: file aliasing is supposed to work with a single disk file.

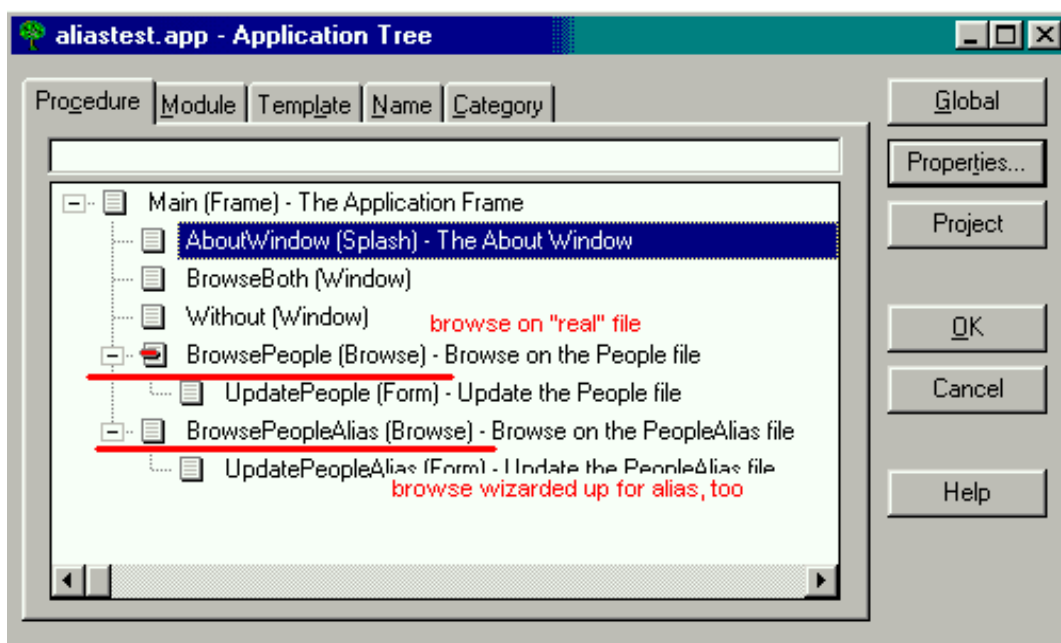
NOTE: While the environment will let you complete the worksheet without specifying a `NAME`, don't.

Finally, given the original purpose of aliases, you will find that the alias inherits the base file's fields and keys but not its relations. You may create any new relations needed, including relations to files already related to the base file. The idea is to use different keys to create new relations.

### What Is An Alias?

With the dictionary complete, if you wizard up an application you will notice something quite remarkable. When you examine the application tree, you will see procedures generated for both the base file and its alias. It is as if the Application Generator is treating the alias as an entirely separate file. Think about this; of course it should!

Figure 6. Automatically generated procedures both "files."



If you compile and run the application, you will be able to call both browses and their update forms without incident. Well, of course you should!

NOTE: Do consider checking the Do Not Populate box for any aliases in your dictionary.

If you examine the generated source for the attached sample application, the impression that there are two entirely different files will be further reinforced. Both files are declared separately and without apparent cross-reference in the main source module. There is a `FileManager` and a `RelationManager` for each. What you will not find is any obvious connection between the base file and its alias, no reference to the fact that one is the "real" file and one (a sort) of pseudonym. Alias is not a Clarion statement, neither is it an attribute like `NAME` or `CREATE`.

If you search all the source files generated (or read the on-line help on the `AliasedFile FileManager` method), you will find (`appNamebc0.clw`) a single reference to the alias:

```
SELF.AliasedFile &= Access:People
```

in the `Hide:Access:PEOPLEAlias.Init` method. By referring the alias' `FileManager` to the base file's `FileManager`, the ABC classes "know" which is the "real" `FileManager` (that is, it does not instantiate a `FileManager` for the alias but uses the base file's `FileManager`). But you do have to dig to find this.

What is less difficult to miss (though it does not jump out and bite your...nose) is that both file declarations have identical `NAME` attributes. And this is the key to comprehending what an alias is, how it works and what it does.

## Built On The Basics

If you examine the code in the main source module, there are three Clarion key words to consider. Combined, they make file aliases work.

First, you will find a `LABEL` for each of the files. In the sample application, for example, you will find both `PEOPLE` and `PEOPLEALIAS`. Different `LABELs` allow you to refer to each of them separately (that's what `LABELs` do).

Next, you will find matching and identical `RECORD` structures. According to the Language Reference Manual, a `RECORD` structure is what creates the memory buffer (per thread, if you select the `THREAD` attribute for the file – see sidebar). And, since there are two `RECORD` structures, with unique prefixes, there will be two buffers. This is what will allow two different records to be in memory simultaneously.

Finally, the same `NAME` attribute is used for both file declarations. Therefore, both `RECORD` structures/buffers refer to the same physical file on disk. And that is what makes an alias work (well, that and the use of the same `FileManager`<sup>3</sup>).

To reiterate, declaring a file alias creates both a second `LABEL` and a second `RECORD` structure.

These are logical entities: like an old-fashioned line number, a `LABEL` identifies a location in the code and a `RECORD` is a (dynamic) location memory.

If you run the sample application, you will see that is entirely possible to populate two browses of the same file on a single window without using aliases. It is possible to do this using different keys for each list (select `Browse|Browse Both -- Without Alias` from the main menu of the example application).

However, no matter what you do, only one record will be in memory. If you run the sample lookup procedure and select a record, you will overwrite the values in the initial record to those of the one selected in the look up. There is no way to look up a supervisor from employee data entry or to select kit items here. Try it, you won't like it.

On the other hand, if you populate a browse from the primary file and a browse from an alias of it, you can have two different records in memory at the same time. Having two records in memory is, of course, not especially noteworthy. What is noteworthy is that both are from the same file, at the same time, on the same thread.

In the sample app, select `Browse|Browse Both` from the main menu. For each browse box there is a hot field indicating the record in memory for each list. Notice that they are only the same if you intentionally select the same record from each list. Run the matching lookup demo and you will see that it operates just as you expect, just as if there really were two copies of the file.

"Buffer" is an extremely important concept in Clarion.

A `Record` buffer is a memory structure which contains values from the currently accessed file record. Think of it as paralleling the file's `Record` structure.

A `Record` buffer is created for each file in a procedure when the file is opened. Thus, record (file) buffers are thread specific.

Further, the buffer only contains "fields" for those fields actually referenced in the procedure. In a Form or Process procedure, this is all of the fields. In a browse, it is not. The buffer in a browse contains only those fields populated in the View. Thus, fields in the list box and fields added to the "hot" list contain current data. Any other file field is unreliable.



## Using An Alias

An aliased file, of course, can be populated and used in exactly same way as any other file. After all, it has a `FileManager` (a reference to the `FileManager` for the base file) and `NAMES` a disk file. Thus, the "real" file will be affected by adds, changes and deletes. Even the Wizards and Wizatrions know this.

But that is not what aliases are about.

The special need addressed by the second buffer that an alias provides is when you need to access the same data set twice on a single thread (since buffers are allocated on the thread; again, see the lookup examples in the demo application).

Say, for example, you have an inventory item that is a kit/bill of materials. On its update form you would like a tab with a browse box listing the component items. These component items may well be stored in a child file but their descriptions will most likely be in the inventory file. So, you need to be in the inventory file twice. The first access gets the data for the record being updated and the second gets the descriptions for the component items.

The browse box descriptions for the kit components would be populated from the aliased file (the alias being related to the child as a second parent). The primary record buffer will still contain the data from the inventory record being updated. Neat. No mashed buffers (which is exactly what happens if you use the inventory file directly for both purposes).

Similarly, if you have an employee record in access, a lookup procedure for the supervisor would be created from an alias for the employee file. Again, two different records from the same file in memory simultaneously with no cross talk on the thread.

## Caveats

In reviewing an early draft of this article, David Bayliss pointed out that what I discuss here is the "proper" use of file aliases.

Prior to C5B, an alias is always required if you access a file twice on the same thread (not simply in the same procedure) for any reason. He provided the following illustration:

An Employee form has a lookup to Department. The browse on Department also looks up the supervisor name (an employee). There are two accesses to Employee here. They are in different procedures but on the same thread. Because they are on the same thread, they use the same buffer (again, see the [sidebar](#)). Prior to C5B, either template set would have problems with this scenario. You could not reuse an existing Department browse.

Since then, intelligence has been added in C5B to try to detect this. Buffers are automatically saved and restored in many cases like this. Specifically, file usage in procedures is monitored (based upon the `UseFile` method). If a child procedure re-uses a file used by a parent and if the child doesn't explicitly say it wants to overwrite the parent record, `Save/Restore` file is placed around the child usage. David Bayliss has prepared an article on this subject ("[Propitious Memory Corruption](#)") which appears in the next issue.

## Summary

If you need to access a file twice on a single thread and those accesses may involve different records, the conventional wisdom is to create and use an alias.

Recursive lookups are the obvious example of needing to access two different records in a single file simultaneously. The other "standard" case is when you need two different relations between the same two files. In these cases, an alias is appropriate and will always work and



failing to use one will cause your app to fail.

If you absolutely insist on not using aliases, again see [David Bayliss's article](#) which describes the enhancements to USEFILE.

[Download the example application.](#)

---

[Steve Parker](#) started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitor's right side mirrors -- while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.

In This Issue

[Alias - Who Was That Masked File?](#)

Posted on August 24, 1999

[Open Source Products Update](#)

Posted on August 24, 1999

## Notes

<sup>1</sup> The Relational Model also frowns on this practice. But, in the words of Randy Goodhew, the nice thing about the Relational Model is that applications following it rigorously are unusable. Reality often conflicts with theory and, in such cases, the nod tends to go with reality (at least among Topspeeders).

<sup>2</sup> If you are not familiar with this term from the pre-PC era, it refers to a programmer's ability to successfully merge geometrically divergent shapes and/or expeditiously scale vertically enhanced edifices (putting square pegs into round holes and leaping tall buildings in a single bound, doof).

<sup>3</sup> Of course, you can cut and paste a file declaration, changing the PREFIX, but use the same NAME. You will be able to access the single file from two browses and do so simultaneously. You will be able to update the file from multiple browses simultaneously. You will be able to use one as a lookup file for the other. Ok, so you instantiate a second FileManager. But other than that, what's the big deal?-Clever, eh? Open two browses on a single file created in this way. Change an existing record in one of the browses. Now, highlight the "same" record in the second browse and press the Change button. It isn't updated and, in fact, it won't until you save your edits. So, the big deal is that cleverness can result in lost concurrency checking.

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).



# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## The Clarion Advisor

### Changing An Application's Dictionary

by Mike Pickus

Have you ever wanted to import a procedure from another application and found it was created with a different dictionary? It happened to me recently, only I needed integrate six applications into an EXE with DLLs and each application had a slightly different dictionary. When I tried to change the first application's dictionary, I saw the following warning:

"This is only guaranteed to work if the dictionaries are the same. It is not sufficient to have the same files and fields."

Sounds ominous. Why isn't it sufficient to have the same files and fields? Because every file, key, and field in a Clarion application uses a pointer (a unique identifier called an IDENT) to a corresponding file, key, or field in the dictionary. That's why when you change the name of a field in your dictionary, your application still works.

When you change the dictionary of an application the IDENTs won't match unless the dictionary is the same. The trick is to remove the IDENTs so that matching is done by file/field/key names. The following steps and program should enable you to change to a dictionary that has all of the same files, keys, and fields, but different IDENTs.

1. Ensure that all of the files, keys, fields, and relationships in the application are also in the target dictionary. If the differences are minor, you can open both dictionaries and copy (Ctrl-C) and paste (Ctrl-V) the files from a source dictionary to target dictionary. If they are substantially different, you may want to export the source dictionary to a .TXD file (File|Export Text) and import the .TXD file into the target dictionary (File|Import Text). If you have the Enterprise Edition, you also have the option to use the synchronizer. Save the dictionaries.
2. Open the application and export it to a text file. File|Export Text creates a AppName.TXA file.
3. Compile the DeleteIDENTs program, shown in Listing 1 (the source and project are available for [download](#) – if you create your own project to go along with Listing 1 make sure you include the ASCII file driver). Run DeleteIDENTs to convert the AppName.TXA file into a new file, AppName\_NoIdents.TXA, with the IDENTs removed.
4. Create a new application. Select the target dictionary (Dictionary File) and delete the "MAIN" (First Procedure).
5. Import the new .TXA file. (File|Import Text) the AppName\_NoIdents.TXA. When the IDENTs are missing, the import uses the file, field, and key names to match the same names in the dictionary. You now have a new application with a new dictionary.

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

If you're interested  
take our poll &  
let us know!

## Listing 1. The DeleteIDENTS program.

```

PROGRAM

MAP
END

IDENT      UNSIGNED, AUTO      ! INSTRING returns an UNSIGNED
CloseP     UNSIGNED, AUTO

eIDENT     EQUATE('IDENT(')
eCloseP    EQUATE(')')
eNoIDENTs  EQUATE('_NoIDENTs.TXA')
eDict      EQUATE('DICTIONARY')
! eMax varies with the # of relationships
eMax       EQUATE(6144)

OldFileName CSTRING(129), AUTO      ! TXA in
NewFileName CSTRING(129), AUTO      ! TXA out

OldFile     FILE, DRIVER('ASCII'),NAME(OldFileName)
RECORD      RECORD
aLine       STRING(eMax)
..
NewFile     FILE, DRIVER('ASCII'),NAME(NewFileName),CREATE
RECORD      RECORD
aLine       STRING(eMax)
..

CODE
LOOP WHILE FILEDIALOG('Choose TXA to Convert', |
  OldFileName, 'TXA|*.TXA')
  NewFileName = OldFileName [1 : LEN(OldFileName)-4] |
    & eNoIDENTs
  CREATE(NewFile)
  OPEN (NewFile)
  OPEN (OldFile)
  SET (OldFile)
  LOOP
    NEXT(OldFile)
    IF ERRORCODE(); BREAK.

    ! Don't use the old dictionary
    IF OldFile.aLine [1 : 10] = eDict; CYCLE.

    ! Is IDENT on this line?
    IDENT = INSTRING(eIDENT, OldFile.aLine, 1, 1)
    IF IDENT

      ! Find closing parenthesis
      CloseP = INSTRING(eCloseP, OldFile.aLine, |
        1, IDENT+6)

      ! String slice the offending IDENT
      NewFile.aLine = OldFile.aLine [1 : IDENT-1] & |
        OldFile.aLine [CloseP+2 : eMax]

      ! If there is nothing to save then cycle
      IF LEN(CLIP(NewFile.aLine)) < 5; CYCLE.

```

```
        ELSE

            ! If there is no IDENT, save the whole line
            NewFile.aLine = OldFile.aLine
        END
    APPEND(NewFile)
END
CLOSE (NewFile)
CLOSE (OldFile)
MESSAGE('Done')
END
```

Editing exported dictionary and application text files is a powerful option. For instance, you can globally change all date formats from @d1 to @d2b or even change from the Clarion template class to the ABC template class. But always remember to create new dictionaries and applications rather than overwriting your old ones.

[Download the source code](#)

---

Mike Pickus is a member of Team TopSpeed QA and hosts the VA/MD/DC TopSpeed User Group in McLean, Virginia. His latest project is a touchscreen POS application.

In This Issue

[The Clarion Magazine Technology Poll](#)

Posted on August 31, 1999

[The Clarion Advisor: Changing Dictionaries](#)

Posted on August 31, 1999

[Larry Teames On Reports](#)

Posted on August 31, 1999

[Propitious Memory Corruption](#)

Posted on August 31, 1999

[Clarion News - August 1999](#)

Posted on August 31, 1999

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).



# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## Feature Article

### Reporting With Clarion

by Larry Teames

This column is part of a series in which I illustrate approaches to common reporting situations. To make it easier for you to follow and later review what I cover, I have created a small dictionary (BREAKS.DCT), and the TPS files to go with it (LETTERS.TPS, BREAK1.TPS, and BREAK2.TPS). These files and the application I create for each column are available for [download](#). This month's application file is [CMEX2.APP](#).

I'm using Clarion5 (ABC) to build the example applications, but in most cases, I use the legacy embeds when placing source code in the report procedures because of their more descriptive names.

#### Working with Letterhead stationary

This month I've built a report that allows the user to print a letter (the body text and recipient are selected at runtime) using letterhead stationary for the first page, and plain stock for the remaining pages. I've elected to do this "one recipient at a time" to accommodate the simplest scenario, in which the user places one letterhead page and several stock pages in the printer's paper bin before printing.

Because all letterhead stationary doesn't extend the preprinted portion the same distance down the page, I'll need a way to allow the user to control how far down on the first page printing will begin. However, I want to start printing at the top of each subsequent page.

#### The Approach

First I created a standard ABC report procedure, where I specified the BREAK1 file as the primary file and the LETTERS file as an Other file. I set this procedure up so that the user selects the desired letter text (from the LETTERS file) and then selects the desired recipient (from the BREAK1 file). I also decided to use a simple INI file-based approach to control the distance I need to skip to start printing below the preprinted part of the letterhead stock.

#### Formatting The Report

First I went into the report formatter and deleted the automatically generated Page Header, Page Footer, and Form bands. I don't need these bands for this report, and they will only complicate things by trying to automatically print.

Next I set the width of the report (the Detail Area) to 7½ inches to allow for a ½ inch margin on each side. Then I set the Y-position of the report to ½ inch from the top of the page, and the height to a little over 9½ inches.

Finally, I set the default font for the report to Arial Regular 10pt.

#### Creating And Populating The Bands

*Serious  
Reporting  
Made  
Easy!*

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

If you're interested  
take our poll &  
let us know!

To have this report operate the way I wanted I needed three detail bands, so I created two additional detail bands. I renamed the original band from Detail to SpacerBand, and also gave it a field equate (FEQ) label of ?SpacerBand. I named the second band HeaderBand (with the FEQ ?HeaderBand), and the third band I named Detail (?DetailBand).

SpacerBand will be blank when printed, and is used primarily to push the subsequent bands down below the preprinted part of the letterhead stock. I say primarily, because I also used it as the "home" of the textbox control containing the memo (LET:LetterText) from the LETTERS file. Note that when I populated this control I left it set to default height and width and didn't concern myself with where on the band I placed it. This is because this control will be used indirectly to supply text lines for the letter, but will not actually be printed (I hide it at runtime). This band was a convenient place to put the control where it would not be in my way while working with the other bands and controls.

HeaderBand is used to print the letter recipient's information from the selected BREAK1 record. There's no trickery with this band. It's simply made tall enough to allow for the controls that contain the recipient information.

DetailBand is used to print the text from the memo field. However, this is done one text line at a time. This approach allows me to have the letter automatically flow from one page to the next, regardless of the number of lines entered into the memo. After renaming this band, I populated a STRING control, made it as wide as I wanted each text line to be, then renamed it ?TextLine. Note that this is a simple string control, not a FEQ for a variable. Finally, I changed the text of the string to something that would remind me of its intended use.

### How it works

Now let's look over the code that makes this report work. After selecting the desired LETTERS record and recipient (BREAK1), and opening the report, I set up the report with various runtime changes. In the After Opening Report embed I placed the code in Listing 1 (the comments describe what the code is doing):

Listing 1. The After Opening Report embed code.

```

SETTARGET(REPORT)
! Following FEQs are in the REPORT
! Get the letterhead skip area
IF GETINI('Letter-Standard','SkipForLetterhead','NotFound'|
, '.\CMEX.INI') <> 'NotFound'
  ! Reduce by report's printable area offset
  LHSkip# = INT(GETINI('Letter-Standard','SkipForLetterhead',|
  'NotFound','.\CMEX.INI')) - REPORT{PROP:YPOS}
  ! if result is less than zero set skip height to zero
  IF LHSkip# < 0
    LHSkip# = 0
  END
  ! Set "spacer" detail height to skip height
  ! and maxheight to that same height
  ?SpacerDetail{PROP:HEIGHT} = LHSkip#
  ?SpacerDetail{PROP:MAXHEIGHT} = LHSkip#
END
! Set textbox width to fixed string length
?LET:LetterText{PROP:WIDTH} = ?TextLine{PROP:WIDTH}
! Ensure the textbox is hidden
?LET:LetterText{PROP:HIDE} = True
! Ensure the textbox is set to DEFAULT height
?LET:LetterText{PROP:NOHEIGHT} = True
! Ensure Detail band set to DEFAULT height

```



```
?Detail{PROP:NOHEIGHT} = True
! Ensure string ctl ypos set to zero
! and its height set to same as textbox lines
?TextLine{PROP:YPOS} = 0
?TextLine{PROP:HEIGHT} = ?LET:LetterText{PROP:LINEHEIGHT}
! Reset to default target (ie. window)
SETTARGET
```

After setting these report characteristics, I used the Embeditor to find the generated print statements and omit them from the compile. I then followed the omitted code with custom code to do the printing in the way I needed it, as shown in Listing 2.

Listing 2. Overriding the default print behavior.

```
! Omit the generated print statements
! for all bands print behavior.
OMIT('--omit to here--')
PRINT(RPT:SpacerDetail)
PRINT(RPT:HeaderDetail)
PRINT(RPT:detail)
! Start of "After Printing Detail Section"
! [Priority 4000]
!--omit to here--
! Print the "spacer" band to get past letterhead, logo, etc.
PRINT(RPT:SpacerDetail)
! Print the address, salutation, etc.
PRINT(RPT:HeaderDetail)
! Loop thru all lines of the textbox. For each line
! set the string to the next line of the textbox
! and print that line.
LOOP L# = 1 TO REPORT$?LET:LetterText{PROP:LINECOUNT}
  REPORT$?TextLine{PROP:TEXT} = |
    REPORT$?LET:LetterText{PROP:LINE,L#}
  PRINT(RPT:Detail)
END
```

That's all there is to it! As you can see, there wasn't much to code once I decided what I wanted to do and decided how I would need to approach it.

As I've stated in past articles, the best way to control the report engine is to eliminate any undesirable automatic print engine functionality whenever possible. In this example the key was to not print the memo in a textbox (which is the approach typically used). Instead, I broke the memo down into individually printed lines of text so I would get automatic page overflow as each page is filled.

If you have a particular reporting question or problem that you'd like to see covered in a future column, please email me at [lteams@cpcs-inc.com](mailto:lteams@cpcs-inc.com) with your request. I can't promise that I'll be able to use every request I receive, but I'll try my best.

Until next month, Happy Reporting!

[Download the example application](#)

---

Larry Teames is an independent software developer, and one of the four founding members of Team TopSpeed. He is also president of [Creative PC Solutions, Inc.](#) which markets the popular Clarion 3rd party product

In This Issue

[The Clarion Magazine Technology Poll](#)  
Posted on August 31, 1999

[The Clarion Advisor: Changing Dictionaries](#)  
Posted on August 31, 1999

[Larry Teames On Reports](#)  
Posted on August 31, 1999

[Propitious Memory Corruption](#)  
Posted on August 31, 1999

Creative Reporting Tools.

[Clarion News -  
August 1999](#)  
Posted on August  
31, 1999

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).



# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## Propitious Memory Corruption

by David Bayliss

(This article is the second in a series. In the [first article](#) Steve Parker examines the "correct" use of file aliases. In this second article David Bayliss explains the theory and practice of file aliases in C5 ABC. ed)

I clearly remember my first introduction to the Clarion application paradigm. It was during the attempted closedown of the CFD 3 beta: the Florida R&D department was having trouble tracking down some problems with the templates and they sent them over to see if I had any ideas. They sent over a note: "Any ideas why this doesn't quite work?" I responded: "I have loads of ideas why it doesn't quite work, what I don't get is how it works at all...."

To understand my response you need to understand my training. I came to computer science from pure mathematics, where my primary interest was computer languages. As such I had studied many of them, the distinctions between them, their strengths and weaknesses.

One of the main issues that we studied was minimisation of scope. The logic is extremely simple. Programs work best when all the variables have the values they are supposed to have. Thus good program design maximises the chance of each variable having the right value. You maximise the chance of a variable having the right value by minimising the chance that someone gives it the wrong value. You do that by stopping them touching it, and you stop them touching it by making sure it isn't declared where they can see it.

The simple rule to implement the above is avoid global variables. The more modern and OOPy way is to use the `PRIVATE` attribute as often as possible. In a language such as C++ you can go further and declare variables halfway through a procedure to stop people further up the procedure touching it. You can even define nested scopes so that a given variable is only visible over (say) three lines of a procedure.

Another key feature of safe languages is the ability to clearly define an interface, especially between procedures. In particular you minimise the number of procedures with side effects. Ideally you use functions that return results and if you want the result again you re-call it. The ultimate is the "provable" language where you define what a procedure does purely by the incoming parameters.

Of course all of these heuristics are made to be broken, and where required one would slip in a global variable, or allow some procedure to have an external side effect. But each time I did so I was mindful that I was reducing the maintainability of the app, reducing my ability to scale the app and ultimately reducing my productivity.

Now the Clarion system has a proven ability to produce massive, maintainable apps, astonishingly quickly. Studying the legacy application paradigm was something I looked forward to.

### Clarion Legacy Application Paradigm

What I saw took my breath away. The Clarion application paradigm is simple, audacious and astonishingly effective. Everyone just shoves

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

If you're interested  
take our poll &  
let us know!

values into global variables (usually file buffers), assumes everyone else has put just the right value into just the right variable at just the right time and carries on regardless. And it works, very very well.

In a browse a given variable could easily be used for six or seven different logical things. It could be the use variable of a hot field, a range-limit field in a filter, a locator field, the use variable for the locator entry control, a parent range-limit of a child browse, the selection field for a vertical thumb, the source for computed fields and the scratchpad I use to load the data before copying it into the queue.

Most germane to this article is the behaviour across a procedure call. When a browse calls an update it just reloads the present record and calls the update form. When the update form returns it just assumes the record coming back is a good one, repositions itself and keeps going.

The "current value" range limit works similarly; it locks the browse to be range limited by whatever happens to be in the record buffer when the procedure is entered.

Selection from a browse again works simply. The browse just "leaves the record buffer" on the current record (every time you change the selected item the record buffer changes) and the update form uses any field it feels like.

So why didn't the templates quite work? Well the first problem is the "six or seven logical things" I mentioned above. I suspect that as you read them you thought, "they're all the same!" And they are usually, but not all of them all the time.

Consider the locator field when you have type in "fred" and the selected record is "gerry" (fred not existing), what should the locator field show? What should the attached entry read? What is the value of that field if it is also used as a hot field? As a range limit on a child browse?

Suppose I enter a filter that renders the browse empty. Is the field now blank? Even if it is a range limit field?

How about when I am loading in data to fill the browse; do I reset the field used as the range-limit for the child browse? Or do I wait to the end?

## The Cost

Over many legacy template releases these questions were all answered to the point where buffer variables generally had what most people wanted in them most of the time, but there was a cost. Every time a logical part of the browse used a global variable it also had to post a message to the browse to refresh itself so that the expected logical value was in the corresponding global variable. The affect was that legacy browses always loaded the file data at least three times, and you could concoct examples (using child browses, locators etc) where it would re-load eight times for one notional "refresh".

Another cost is that all the components felt they had to keep the buffers fully loaded. So a browse loaded a whole record for every new selection. A select browse did a full re-load (including child files) upon returning.

So Clarion had successfully married together productivity and functionality by applying a huge dollop of pragmatism.

## The Kids

In many happy marriages there comes a point where the normal peace and harmony is shattered by the arrival of children, and nothing is ever quite the same again. This was true for the Clarion marriage too. CFD made a gigantic leap forward by the introduction of referential integrity (RI) checking. You could now add and delete records from your data and your data would remain consistent, and it happened automatically. The dictionary editor/appgen were extended to understand the idea of relationships, especially the one-to-many relationship.

This technology was vital to the long term survival of the product:

relational database programming had now arrived in a totally safe and automated way.

Usually.

The problem with adding RI to the templates (and other things such as "must be in file" validation) is that suddenly an innocent action upon one file didn't simply change the fields in that file: it could change the fields in any related file too!

Try the following with the legacy templates. Give yourself a many-to-one relationship between two files with cascade on updates and clear on delete. On the update for the "many" file give yourself a field lookup button to a browse on the "one" file with a select button. Enter some "many" records each with "one" lookups (you will need to enter some values to look up). Now edit a "many" record (note the form contents), call up the lookup browse with the field lookup button and instead of selecting a record, delete one. Then cancel the select. You will find the "many" record has magically changed values! If you then press OK (not advised) your data is corrupted.

You can get similar effects with "must be in file" validation upon a form.

The simplest way to avoid most of these problems is to prevent insert/change/delete on select browses (the legacy wizards do this). The legacy RI can still scramble your data other ways but it is far rarer.

## The Rope

With CW 1.5 we added another vital piece of technology that extended the relational nature of the Clarion language: the VIEW. The VIEW is brilliant. You simply declare the primary file, the fields you want from the secondary files and it handles all the rest for you. I genuinely believe the VIEW makes Clarion one of the simplest DBMS systems to use in existence.

Regrettably it makes it a little too easy to use. Whereas before people were quite content to have a few foreign ID fields floating around in their browses there was now no excuse. Simply populate the field from the child file and the VIEW handled the rest. The problem is that whereas a "book browse" would typically alter the "book" file it could now be loading values into subject, author, and publisher files without a moment's thought on the part of the programmer.

Consider a book file related to subject (many-to-one). The subject form would thus have a child tab listing all the books on a particular subject. Now the book browse (on the child tab) would typically have a SubjectId field. This you remove and populate the subject name instead. Seems reasonable and looks reasonable and under legacy it will work...often. There are a few interesting quirks. Firstly, if you go to the child browse and delete all the children when you come back to the General tab (with the form information) all the fields will be blank. If you press OK you lose your parent data. The deadlier one is that if you edit one of the child records and on the child record alter the parent (i.e. you are moving it from one parent to another) when you come back to the form the data you are looking at will be correct but the record under the form will have switched to the new parent! Thus any edits you make to the parent record you can see will actually happen to the new parent of your ex-child!

## The Call Tree

A subtler side effect of the rope given in 1.5 is that grandchildren started to become treacherous. Imagine I am editing a patient file. I have a lookup to the doctor browse, and from the doctor browse I call up the doctor update form to see if a given doctor can do such and such. The doctor update has a child tab of patients which upon loading writes changes all over the patient file. I cancel out of the doctor update, select the suitable doctor and keep going. When I press the OK button I either corrupt my data (legacy) or get an assertion (ABC).

Most people are aware that if they see the \*\*\*Recursive call message on

the procedure call tree then they have a potential problem. Fewer people are aware that if ever any file appears twice in a procedure call tree (other than by design) then mayhem may ensue. The recursive warning is just a guarantee things are going to go wrong.

## Then ABC Came Along And It All Stopped Working!

At least that is what I have been told by many people, at great volume. ABC did make some fundamental changes but I believe they are justifiable. I'll go through the four cases in sequence:

1. Inefficiency due to multiple usages of a variable within a procedure, especially the browse. This has been tackled in ABC by teasing out some of these logical uses, so locators have their own shadow, range-limits are stored by value, etc. By doing this ABC only ever loads data once, it avoids painful `REGETs`, and generally is more efficient under client server and especially SQL. The down side is that hot fields have to be defined as hot fields and reset fields as reset fields (previously you could forget to fill them in and generally got lucky). Further, from a select browse you can only actually guarantee the linking field will be filled in (although C5 has a `SelectWholeRecord` property to force a full record reload upon selection).
2. Unexpected record corruption due to automatic file validation and RI. ABC has eliminated these problems using save/restore file technology. There usually isn't a down-side unless someone was using one of these corruptions to good effect.
3. Use of a file twice within a procedure. This is the one people don't like. In legacy this will usually appear to work, even when it doesn't work the form looks okay, it is just the data on disk that gets corrupted. ABC is rather different. In this situation it will Assert, throw up garbage screen displays (if the assertions are ignored), and generally make it clear something is horribly wrong. I have had many people ask "Why can't you get it to work like it did in legacy?" They never believe that legacy doesn't work until I take them through the steps. So what is better? A system that works 99% of the time and just subtly corrupts your customers data or a system that downs tools and refuses to budge until it is fixed? See some of my "[offensive programming](#)" articles for the line I take.
4. Up until C5B ABC treated case 4 much like case 3. If the user dug far enough down the procedure tree to cause an unintended corruption then upon returning to the grandparent procedure it Asserts. Again I claim this is better than the "corrupt and continue" approach. However, it isn't quite as nice as case 3 because there is nothing to guarantee that the procedure tree will be fully "drilled down" during testing.

The only real solution to 3 and 4 is what I call "structural" aliases. [Steve's article](#) deals with the use of aliases when there is a logical reason for them. Cases 3 and 4 require that sometimes you need to use aliases even when they are representing the same logical entity.

## Automatic Aliases

C5B makes a slightly radical but very strategic step towards solving case 4. Essentially the idea is this each procedure declares which files it is using, then if it calls a procedure that uses the same file the child will save the contents on entry and restore them upon exit so that the parent procedure has the record buffer it was expecting. This is managed by a global "FilesManager," so if a great-great grandchild starts using a file it is again saved/restored as required. You can even go in and out of recursive procedures and everything still works. Note that the child does not get a cleared record buffer, so data transmission from parent to child still happens as before.

Of course this is not quite enough. Sometimes a child wants to change the data so that a parent can use the result. To enable this to be specified (and for the procedure to register the files it uses) the `UseFile`



method was adapted to take a `UseType`. These are as follows:

- `Corrupts` – This file will be altered by this procedure but the procedure is not bothered about the contents of the buffer across procedure calls.
- `Uses` – This file is altered by this procedure but expects its children not to corrupt the buffer across procedure calls.
- `Returns` – This file will be altered by the procedure and the alterations should be transmitted back to the parent (note this overrides the `Uses` declaration in the parent).

The ABC templates have also been adapted so that the various control templates know the action they are supposed to have upon a file and make the `UseFile` calls accordingly. ABC template applications thus work unchanged from C5A to C5B. However hand-code (or non-TopSpeed templates) may need altering to register that a given child wishes to change a file buffer used by a parent.

### In Conclusion

There is no such thing as a free lunch. The Clarion application paradigm has essentially bucked the trend for many years to deliver productivity and functionality. ABC has eliminated two of the compromises made by the legacy templates to achieve this. The fourth has now largely been tackled by C5B at the cost of some code compatibility. ABC is presently unable to tackle the third issue although it does flag the error enabling the user to take action.

---

[David Bayliss](#) is a Software Development Manager for TopSpeed Corporation. He is also TopSpeed's compiler writer and the chief architect of the Application Builder Classes.

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).

In This Issue

[The Clarion Magazine Technology Poll](#)  
Posted on August 31, 1999

[The Clarion Advisor: Changing Dictionaries](#)  
Posted on August 31, 1999

[Larry Teames On Reports](#)  
Posted on August 31, 1999

[Propitious Memory Corruption](#)  
Posted on August 31, 1999

[Clarion News - August 1999](#)  
Posted on August 31, 1999



# Clarion magazine

Main Page
Log In
Subscribe
Open Source
Links
Mailing Lists
Advertising
Submissions
Contact Us
Site Index
ClarionMag FAQ
Download PDFs
Search ClarionMag

## Clarion News

August 31, 1999

### [New Third Party Products From CHS](#)

CHS Associates has released several Clarion add-on products including an application message utility which allows standard application messages to be kept in a configurable database, a calendar lookup with date calculation and personal organizer, and a code tips knowledge base for your development team. Prices range from £19.99 to £59.99.

### [Batch Compiler Source Available](#)

Ralf Schoeffler has made his CWCM Clarion Compile Manager freeware – source is also available. Also at this web site: an MS SQL create script utility and a calendar TXA.

### [ABC Free Templates and Tools 2.0](#)

The ABC Free Templates and Tools 2.0 have been updated with a modification to the FTP class and a new Windows registry class.

### [Registry Function Library 1.1 Now Available](#)

Registry Function Library 1.1 is now available for download. Changes include added features for GetReg, PutReg, and ImportRegFile functions, and bug fixes. Source code is now provided instead of LIBs.

August 24, 1999

### [Database Replication Info](#)

Jerry Ray has a new location for his database replication site. The site contains a lot of information on replication including Jerry's DevCon '98 presentation files, example app, synchronizer source code, and DUN/RAS connection code. There are also links to other replication information.

### [ForKeeps 4.01 Available](#)

ForKeeps 4.01 is now available. This release fixes several bugs in Outlook Express importing. This upgrade only affects people importing from OE5 or from Outlook via the Task Manager.

### [CapeSoft MakeOver Update](#)

CapeSoft is holding a challenge for Makeover users. Submit a Makeover theme, and if they like it, they'll include it in the Makeover package and give you a CapeSoft product of your choice. Also new is MakeOver 1.4. This release includes improved support for NT4 SP4 & SP5 and better font support.

### [Special Agent 1.18 Released](#)

This release of Special Agent from CapeSoft clears up some minor internal bugs with multi-DLL applications.

### [Accounting Package with Source Code Available](#)

KV Enterprises Inc. has the following modules available to Clarion Developers: Accounts Receivable, Account Payable, General Ledger, Inventory Control, Purchasing, Order Entry, and Payroll. The price is \$399 per module, or buy five, get two free. A demo is available.

Great Opportunity!  
Salary to \$125,000+  
Pre-IPO Stock Opts  
Sunny Boca Raton

If you're interested  
take our poll &  
let us know!

In This Issue

[The Clarion  
Advisor:  
Changing  
Dictionaries](#)

Posted on August  
31, 1999

[Larry Teames On  
Reports](#)

Posted on August  
31, 1999

[Propitious  
Memory  
Corruption](#)

Posted on August  
31, 1999

[Clarion News -  
August 1999](#)

Posted on August  
31, 1999

### [LogFlash For Legacy \(Clarion\) Templates](#)

LogFlash is now available for Legacy templates as well as for ABC. This is an activity logging template which provides an audit trail and undo capabilities for database updates. Records can be undeleted, and individual fields can be rolled back. All source code is supplied. Demo available.

### [Gitano One Day Sale Aug 27<sup>th</sup>](#)

Gitano's Jesus Moreno is 40 years old, and to celebrate Gitano is having a one day sale on Friday, Aug 27<sup>th</sup>. Buy one GReg, GCal, GCalc, GNotes, GBuddy, GCal/GCalc Bundle, or GCal/GCalc/GNotes Bundle and get a 40% discount on one of GReg, GCal, GCalc, or GNotes. (Hey, Cowboy, why didn't you have a sale on your 40<sup>th</sup> birthday? ed.)

August 17, 1999

### [Internet Framework Client Templates](#)

THE IFT-HTTP Client templates have been added to the Internet Framework template family. These templates are for apps that can use the web to stream data and upload/download binary files and text. They can be used with IFT-built web servers or Clarion Web Edition products. Introductory price is \$89.

### [Memory Manager Source Available](#)

In 1995 Randy Goodhew wrote a 16 bit memory manager for Windows 3.1 called RESORZ. The program was written in Clarion with a little assembly language. RESORZ will display all memory blocks in use, hex view the memory space of other processes, allow you to terminate a stuck task, compact 16 bit memory, install guard blocks that protect low memory against task hogging, display resource usage and the number of unused file handles, monitor disk usage in background, and more. This is a terrific example of how the Clarion language can be used to write system level utilities.

### [TimeSavers Scheduler Source Code And Template](#)

Positive Software has announced its TimeSavers scheduler product which includes day view and month-at-a-glance calendar view. Appointments can be dragged/dropped and the appointment queue can be quickly customized. A demo is available, and final release is expected August 27, 1999. Price is \$79 until August 27<sup>th</sup>, \$99 thereafter.

### [VCS Difference Reports](#)

An alpha build of the VCS Difference Reporter is available for download. It requires Clarion 5b or later and VCS 2.02. You can generate difference reports on source to source comparisons (including APPs) as well as source to archive and archive to archive.

### [C5 game from Gitano Software](#)

New from Gitano is the game of d a d o s. This game is similar in play to Yahtzee Requires C5b runtime dlls. Beta shareware, with a nag screen.

### [DevCon Early Registration Extension](#)

Early registration for DevCon has been extended to August 20<sup>th</sup>, after which conference registration goes up by \$100. Register on-line or call TopSpeed Sales at (800) 354-5444 or (954) 785-4555.

### [G-Buddy In Pre-Release](#)

G-Buddy is a collection of five applets: Message Buddy, Note Buddy, Image Buddy, Color Buddy, and Template Buddy. Three of the five are available in pre-release. Price is \$69 before final release, \$99 after.

August 10, 1999

### [Topspeed Newsletter](#)

Topspeed's developer newsletter is available online, and if you wish you can also subscribe to have issues sent to you by email.

### [New Pea Brain Software Web Site](#)

Pea Brain Software's new web site contains information on Data Modeler

for Clarion 5 and other Pea Brain products, including Localizer, an application translator.

#### [Topspeed Publishes ABC E-Book](#)

Topspeed has published a PDF called Learning Your ABCs: Making a Smooth Transition from Legacy to ABC. Includes many comparisons of Legacy and ABC embed points. Example code is also available.

#### [SQL Bill Of Materials](#)

Michael Gorman has posted for download a software application called SQLBOM, for SQL Bill of Materials. The zip includes a user's guide and the "Great News, The Relational Model Is Dead" paper and presentation. Look in the Software Downloads subsection of Free Downloads.

#### [CapeSoft Third Party Initiative/Product Updates](#)

CapeSoft is working toward a standard for third party product installation programs, including use of the Accessories menu in the IDE, standard product directories, template registration, and more. Contact CapeSoft if you have an interest in the proposed standard. Product updates are also available for Special Agent (v 1.17) and FileManager (v 2.8). Heads Up Multi-Proj 2.0 is now in late beta.

#### [Xplore For C5 Updated](#)

Xplore for C5 has been updated with browse filtering based on selected cell contents. As well, a class property can be set so that all sort orders can inherit any new filter. Xplore includes English, German, Dutch and Spanish translation (TRN) files.

#### [New Gitano Web Site](#)

Gitano software has moved its web site to [www.GitanoSoftware.com](http://www.GitanoSoftware.com), which includes a new on-line store. The 20th sale from Gitano's store will receive either GCal or GCalc free. GReg Plus, GCal, GCalc and Gnotes are now all available for C5B.

#### [New In Back Version Available](#)

In-Back v1.504 is now available. This product (written in Clarion) does spot backups of specified files, creating a new archive each time. In-Back uses the LSPack compression library and is available on a 30 day trial.

#### [ForKeeps 4.0 Released](#)

A new release of ForKeeps, a highly-regarded mail and newsgroup archiver written in Clarion, is now available. Version 4.0 supports Outlook Express 5, Outlook (no relation to Outlook Express, available in 32 bit versions only), better generic mail format handling, and import redirection.

---

[Read the July 1999 News](#)  
[Read the September 1999 News](#)

Do you have a news story or press release we should know about? Send it to [editor@clarionmag.com](mailto:editor@clarionmag.com)

Copyright © 1999 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. If you find this page on a site other than [www.clarionmag.com](http://www.clarionmag.com), email [covecomm@mbnet.mb.ca](mailto:covecomm@mbnet.mb.ca).