

## [Clarion Magazine](#)



### [ClarionMag Office Closed May 1-5](#)

The Clarion Magazine office will be closed May 1-5, owing to a road trip lasting longer than expected.

*Posted Thursday, May 01, 2003*

### [PDF For April, 2003](#)

All Clarion Magazine articles for April, 2003 in PDF format.

*Posted Friday, May 09, 2003*

### [New! ClarionMag Third Party Product Directory](#)

Clarion Magazine now has a categorized third party product directory! If you're the owner/maintainer of any Clarion-related product or resource, commercial or free, you can maintain your own product listings here! Your additions/changes are posted immediately. Besides the usual product types, there are categories for included components and compatibility, so you can easily find, say, all C6 compatible products, or those which come with source code. The directory has its own tab, so for ready access just remember to click on PRODUCTS!

*Posted Friday, May 09, 2003*

### [My Experiences With The CHT Server Templates](#)

Can a Clarion programmer with little or no Internet development experience really put an application up on the Web with little effort? Gene Witherup discusses his experiences with the Clarion Handy Tools server templates.

*Posted Tuesday, May 13, 2003*

### [More Macros](#)

Macros are one of the least-talked about features in the Clarion IDE, probably because until Clarion 5.5 you couldn't save them. But you can, and macros are a great aid to productivity. Tom Giles shares some of his favorites.

*Posted Thursday, May 15, 2003*

### [XML For Clarion Developers](#)

XML is becoming a major factor in business software

Clarion Magazine **subscriptions** (online only) are \$49 for six months, \$95 for one year and \$170 for two years.

[Subscribe](#) [Renew](#)

## [News](#)

[EasyExcel Special Offer](#)

[CapeSoft Draw 2 Goes Gold](#)

[Free Parallel Port Template](#)

[EasyListPrint Review](#)

[SimTabTree Template Special](#)

[Clarion 6 Early Access Release 4-3](#)

[New Version Of ABCFree Templates And Tools](#)

[EasyResizeAndSplit 1.08](#)

[cpTracker 50% Off Sale](#)

[Riebens Systems Ends Product Development](#)

[Clarion SQL Seminar](#)

[ImageEx And SysTrack Special](#)

[Gitano 50% Off Memorial Day Sale](#)

[Lindersoft WebStore Officially Launched](#)

[Lindersoft Vacation Notice](#)

development. But what is XML, and what options for reading/writing XML are available to Clarion developers?

*Posted Friday, May 16, 2003*

### **Weekly PDF For May 11-17, 2003**

All ClarionMag articles for May 11-17, 2003 in PDF format.

*Posted Tuesday, May 20, 2003*

### **Write For Clarion Magazine**

Clarion Magazine is looking for writers! If you have something of interest to Clarion developers, consider writing an article. You don't need prior writing experience, just a willingness to learn and share your knowledge with others.

*Posted Wednesday, May 21, 2003*

### **Veronica's Short History Of The Windows Operating Systems (Part 1)**

It is true that the Clarion environment tends to, or tries to, hide as much of Microsoft's OS as possible. This is good for beginners, but bad for the experienced programmer. Sooner or later beginners (if they don't become discouraged) gain a certain amount of experience and want more. In this series, Veronica Chapman surveys the history of the Windows operating system(s), and provides an introduction to Clarion's Windows API capabilities. Part 1 of 2.

*Posted Thursday, May 22, 2003*

### **Finding Source With Enhanced Templates**

Although templates are wonderful they can also be the source of great frustration. Something like a small change in the dictionary can create havoc when you try to compile the program associated with that dictionary, and it can be very difficult to locate the offending source. As Steffen Rasmussen shows, some template modifications can make finding that problem source code a lot easier.

*Posted Friday, May 23, 2003*

### **Book Review: Managing & Using MySQL**

This O'Reilly book contains some very good chapters on SQL and MySQL, including database administration and performance tuning. It also contains a lot of information that will only be of interest to a minority of Clarion developers. On balance, a recommended read for those new to SQL and MySQL.

[MySQL Usage Notes](#)

[PD Browse Button Lookup 60-01 Beta](#)

[EasyListPrint 1.04](#)

[CPCS For EA4-2](#)

[Plugware Email Offline For 12 - 14 Days](#)

[ImageEx 2.1](#)

[EasyListPrint 1.03](#)

[ControlMonitor Free Templates](#)

[xDataBackupManager Pro v1.6](#)

[xTipOfDay v1.7](#)

[Data Modeler 6.0 Update](#)

[Logos, Design Banners, Etc.](#)

[Icetips May 2003 Newsletter](#)

[Replicate Version 1 Beta 15](#)

[PDF-XChange Twain/WIA Update](#)

[gReg Update](#)

[Image-XChange SDK Demos/Docs](#)

[CWPlus Review At ClarionShop](#)

[SetupBuilder 5 News](#)

[Scan2PDF Update In PDF-Xchange SDK 2.5](#)

[ClarioNET 1.3 Server Deployment Manager](#)

[Clarion TXA & TXD Scanner FreeWare](#)

[cpTracker Lite Released](#)

[Serial Comm Lib 1.1 New URL](#)

[Fenix Beta Program Filled](#)

[Image-XChange Clarion Templates](#)

[UltraTree Platinum 8 C6 Support](#)

[Vote On ETC-4 Date](#)

*Posted Friday, May 23, 2003*

## **Weekly PDF For May 18-24, 2003**

All ClarionMag articles for May 18-24, 2003 in PDF format.

*Posted Monday, May 26, 2003*

## **Veronica's Short History Of The Windows Operating Systems (Part 2)**

It is true that the Clarion environment tends to, or tries to, hide as much of Microsoft's OS as possible. This is good for beginners, but bad for the experienced programmer. Sooner or later beginners (if they don't become discouraged) gain a certain amount of experience and want more. In this series, Veronica Chapman surveys the history of the Windows operating system(s), and provides an introduction to Clarion's Windows API capabilities, including a nifty example app and an updated LIBMaker. Part 2 of 2.

*Posted Wednesday, May 28, 2003*

## **A Tree In A Page Loaded Browse**

Clarion has a template for building a tree list control (RelTree). One of the disadvantages of this control is that the maximum tree level is fixed by the number of files you use in it. In this article, Ronald van Raaphorst outlines a simple idea for a non-level-limited, page loaded tree, using a standard browse control.

*Posted Friday, May 30, 2003*

Looking for more? Check out the [site index](#), or [search the back issues](#). This site now contains more than 700 articles and a total of over a million words of Clarion-related information.

[RPM & PNet for EA4](#)

[Clarion Training in South Africa](#)

[Search the news archive](#)

Copyright © 1999-2003 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

## [Clarion Magazine](#)



[Topics](#) > [Reviews](#) > [Reviews](#)

## **My Experiences With The CHT Server Templates**

**by Gene Witherup**

Published 2003-05-13

Can a Clarion programmer with little or no Internet development experience really put an application up on the Web with little effort? With the [Clarion Handy Tools](#) you certainly increase your chances!

By way of background, I recently purchased the upgrade to Clarion for Windows (C55ee), and also Gus Creces' The Clarion Handy Tools. I have a DOS background but have been learning Clarion Windows programming, ABC and OOP, and working toward eventual conversion of our in-house developed CPD programs. (That's Clarion Professional Developer, aka DOS for all the Clarion Windows developers.)

When Gus released his build 07B2 in mid-December, I spotted the inclusion of templates for three flavors of a web server: a static web server, an Internet client server, and a full browser server. Gus uses the full browser server as the vehicle for his CHT Newsgroup Server, and includes a complete demo application which shows you how to set up a web-based newsgroup. Nifty! (Actually, what Gus is running might be better called a "discussion group.")

I immediately dug into the newsgroup demo application to see if I could ascertain the capability of this set of templates by looking at the descriptions and data elements that the templates were requesting; the further I got the more excited I became because I had an immediate application in mind! (But I'm getting ahead of myself.)

The first thing my programmer and I did was to modify Gus's static web page demo with our IP addresses to see if we could get a static server up and see it on both sides of the firewall. All we had to do was change the links and we had it

running in a matter of minutes. We now use the static server to allow our owners to privately preview web content prior to releasing it to our web site.

Back to the story. My company is a custom manufacturer, and our orders require regular communication between our Customer Service Representative (CSR) personnel, the sales rep, and the customer. We make use of extensive order notes in our in-house system, and each week our CSR persons print and fax these order notes to the sales reps. Creating a window into our database would be a natural application for the Internet. So we defined the requirement as extracting order information every day on open orders only, including all pertinent notes and shipment tracking information. We would make the web database a subset of our internal database, and refresh it daily.

As I pored over the example newsgroup server, I thought (rather naively!) that I might be able to deploy this application in a matter of a few weeks. Boy was I wrong! One thing about Gus's templates, you can be sure that they are complete and almost bug free. However, you can not always see how to put the pieces together.

I started down a couple different paths. I could not decide if I should start an application from scratch and pull the template extensions in, or if I should take the CHT Newsgroup demo and modify it. As I would become frustrated one way, I would start over with a different approach. Finally, at Gus's suggestion, I chose to use the NG demo and modify it for my own application, thereby changing it from a "web-based newsgroup server" to an "application server".

An internet server is decidedly different than the standard "browses and forms" approach of the Clarion paradigm, but Gus has done a good job of simulating that environment. All of the action occurs in the `ServerWindow` procedure. Once you attach the CHT classes to the `ServerWindow`, that procedure has all of the functionality of a server. The authentication procedures, query procedures, and related processes are predefined through the templates. The `ServerWindow` then calls a process for each browse that the application needs. And there are plenty of embed points so that a programmer can change the standard behavior to suit the application.

The interaction between the `ServerWindow` and the browses, which Gus calls the `BrowserServerHTMLBuilder`, bears some explanation. All of the important interaction of the web server occurs between these two pieces. The `ServerWindow` with its classes and extension templates occurs once in an application. You may then add a **Build Query Page** extension in the

`ServerWindow` for every back end browse that your application needs. The query extension template communicates what the web user wants to see to a back end data retrieval process, the `BrowserServerHTMLBuilder`. This back end process is responsible for retrieving the requested data, and formatting it into a web page. The formatted data is then passed back to the `ServerWindow`, which presents the information to the web user via the `QueryResults` extension template.

The `BrowserServerHTMLBuilder` process is multi-purpose. It performs the back end data retrieval by interpreting the query from the server, formats the data to be presented to the end user, and also handles record changes (inserts, changes, and deletes) via an HTML form. If the application requires only simple forms attached to the browse, then the template handles the form formatting automatically. The developer does not have to worry about the additional skills related to HTML and JavaScript programming.

This is a good time for a digression: Consider data retrieval. The CHT templates are ready for SQL and also handle the other Clarion file drivers. In my application, I am extracting information from the corporate Clarion 2.1 database. This worked transparently with only once small glitch: the Clarion memo field. Apparently a `CLIP( )` on the Clarion 2.1 memo field does not work correctly. After much head scratching, and patient debugging help from Gus, the solution turned out to be quite simple. We defined a `CSTRING` and clipped the memo field to the `CSTRING` before using it in the `HTMLBuilder`. I might add that the small extra cost of the source code turned out to be money well spent because this made debugging much easier.

Now for presentation of the data. Gus has designed the CHT server templates to work with Cascading Style Sheets (CSS) and with an external JavaScript (js) file. This allows the developer to change the presentation and some functions without re-compiling the program. Anyone who regularly checks The Clarion Handy Tools Newsgroup server has seen the result of this, as Gus tweaks the appearance of the pages. For me, this became an opportunity to learn some new and interesting tricks. In many places the templates do not force you into any one approach; for instance, you can define web page text within the program, or you can define it in an external JavaScript file. The choice is yours.

Since I started out to describe my experiences, I have to mention problems I have had with buttons. On a web page, buttons seem to be the primary means of doing things and controlling behavior. The button has to call HTML code or JavaScript to communicate the desired action back to the server. Since I started out with the

Newsgroup demo server to build my application server, I spent more hours than I wish to admit trying to figure out how this communication works so I could modify the buttons to perform functions that I wanted to see happen. In the templates, some buttons are placed on the web page by the design of the template. Others were placed through embeds. The Newsgroup server demo has two browses: messages and members. My application is designed to have a parent browse and two child browses. The parent browse, in turn, is restricted to the authenticated user who sees only his own records. Restricting the parent to the authenticated user is completely easy; handling the child browses was also easy, but not immediately intuitive. I had to work on that one; the problem was related to defining a correct button on the parent browse.

I have our application up and running on an in-house server behind a firewall. Yes, it took me a couple months, but this was mostly work I did "in between my other job." My application is straightforward; it could probably be done as a desktop application in a day using standard Clarion development techniques. Now that I have an understanding of how the server templates work, I believe that I could put the same web application together in a matter of a few days. I had plenty of "trial and error" while I was way down on the learning curve. I must say that the support by Gus Creces was incredible.

In terms of implementation, it is important to understand that the application written with the CHT Browser Server Classes *is* the server. You do not need Microsoft's IIS server or expensive hosting. All you need is a suitable computer running Windows XP and a suitable connection to the internet. This makes for a very inexpensive way to deploy an in-house web server.

I can not close this little story without saying something about the "goodies." Many of you know the Clarion Handy Tools as a series of very helpful templates to aid in Clarion programming. But the server templates are more than icing on the cake; they are like having another whole cake! Not only is this a password-protected single instance server, but it is highly configurable. You can easily send e-mail from within an application, attach ftp services, and do other things I have not even thought about. If you are a Clarion developer and want to put an application on the web in a hurry, then take a look at The Clarion Handy Tools.

Clarion Handy Tools Web Site: <http://www.cwhandy.com/>



## Response from Gus Creces

Gene has done a pretty good job of describing the "Browser Server Technology" as it exists in the CHT DEC 15th 2002 release - Build O7B2.1. And I'm happy with what he was able to achieve. Gene has been more than fair in describing the strengths and weaknesses as he found them. The first and second generation of "Browser Server" were aimed at individuals who have very little, if any, knowledge of HTML and JavaScript and what it takes to get a web page, or web data to display on a page. Build O7B2.1 - the build Gene is using - I would consider to be our "generation 2" product. That product makes web data serving primarily an exercise of your skill with Clarion. Despite his protestations that he's not a Clarion Windows expert, Gene has very solid Clarion skills and the tenacity to stick with problems until he understands and is able to overcome them.

At CHT we're now working hard on preparing "generation 3" of our Browser Server Technology. The main difference in this next release will be the amount of control the developer has over web data page design. Generation 1 and 2 generate the pages for you - using style sheets solely as the means of changing look and feel. The actual HTML is written by the server and sent to the client browser without the developer having to write HTML or use any scripting to make data pages happen. While that makes the whole web interface aspect "easy" in the event you don't know HTML or JavaScript, it also limits page design (Gene's frustration with buttons, for example). So in generation 3 we've added a new layer for those who know page design via HTML and JavaScript. We haven't taken away the "generated" pages aspect of the generation 1 and 2 products, that's still there as before, but we've added extra capabilities for those developers who would be frustrated - design wise - by generated HTML. Generation 3 provides template switches that cause the server to produce JavaScript data objects that are sent to the browser along with pre-designed HTML/JavaScript page scripts written by the developer himself. This allows 100% developer control over page design. It also puts 100% responsibility on the developer to produce valid HTML/JavaScript page designs!

Gene, by his own admission, came into this project having very little internet experience. Rather than a drawback, this was perhaps a blessing since he also had very few pre-conceived notions of how interactive data via a browser should or shouldn't work. If the reader has had a lot of experience with IIS and back end-scripting systems such as ASP or Java or Perl or other such technologies, he/she may be initially misled or disappointed by the fact that



with CHT technology, "back end scripting" is not used. With CHT, back end scripting isn't used, because it isn't necessary. Your back end scripting tool is your Clarion application.

Traditionally, most back end scripting serves two primary purposes: First, back end scripts act as an intermediary layer between the server and the data base. When your server is a Clarion app - as is the case with CHT Browser Server - you can dispense with this layer because Clarion can already read data bases and your server app has a direct connection to the data base via file driver rather than via a scripting layer. Second, back end scripts are also used in one way or another to generate page design, or at least, page components via languages that browser know how to deal with - that is HTML, JavaScript and CSS style sheets. CHT Browser Server can dispense with most of this if you ask for "generated" page designs. In that case your page design is done inside a set of templates that CHT provides. Or you can (in our upcoming generation 3 product) choose to write the HTML/JavaScript/CSS stuff yourself.

---

*Gene Witherup is MIS manager for [Fabtex](#) in Danville, PA. He has been programming in Clarion since 1986. Currently he is learning Clarion for Windows, and is working toward guiding Fabtex into a Windows environment.*

## Reader Comments

[Add a comment](#)

Copyright © 1999-2003 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

## [Clarion Magazine](#)



[Topics](#) > [Tips/Techniques](#) > [Macros](#)

### More Macros

by Tom Giles

Published 2003-05-15

Clarion has a really neat feature that many of you probably don't know about. It is the Macro recorder for the Clarion Text Editor. Basically it allows you to capture a series of keystrokes and save them as a single shortcut key. Macros were available in Clarion 5 but could not be saved from one Clarion session to the next. In Clarion 5.5 they can be saved which makes them very valuable.

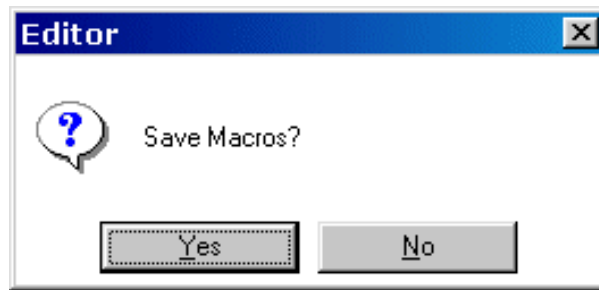
Clarion macros have been written up in Clarion Magazine [before](#), but I'll briefly describe the process before going on to some of my favorite macros.

To set up a macro press **Ctrl-=** (press the **Ctrl** key and = key at the same time) while in the Clarion Text Editor. A window titled "Press Key for Macro" opens (see Figure 1).



**Figure 1. Choosing the macro shortcut key**

Press the key or key combination you want to use as a shortcut and it will be entered in the input field labeled **Key**; then press the **OK** button. Now record your keystrokes, then press the same shortcut key again to stop the action. The system will beep to indicate recording is stopped. In Clarion 5.5 when you leave the Clarion environment you will be asked if you want to save your macros (see Figure 2). Just click on the **Yes** button. Now they will be available for your next session.



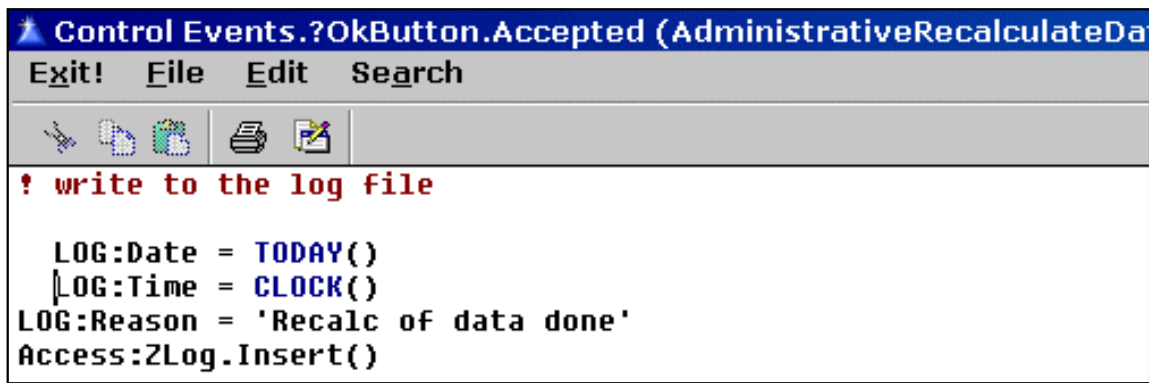
**Figure 2. Saving the macro**

I find I end up writing a lot of code in various embed points for error checking or to do something special. I also like my code indented and formatted in a clear and concise manner to make it easy to read. I also believe in a lot of comments which is invaluable when you come back to that code in six weeks or six months and want to understand it. I have one program still in use that is 20 years old (originally interpreted Microsoft BASIC, later compiled), so good, clear comments and code are essential.

Here are a couple of examples or suggestions that will apply to almost everyone. Get in the Clarion Editor at any source (embed) code point. Press **Ctrl-=**, then the function key **F2** then click the OK button. You are now ready to record your macro. Press the following keys in sequence

Home  
 Home  
 SpaceBar  
 SpaceBar  
 DnArrow  
 F2

Your computer will beep to indicate macro recording is complete. Now put the cursor on any line of code and press **F2** several times. Note the cursor jumps to the beginning of the line (column 1), then moves the text two spaces to the right, then drops down one line so it is ready for the next. By placing the cursor on the first line and pressing **F2** several times each will be indented two spaces (see Figure 3). This is easier and quicker than dragging the series of lines to highlight them, clicking on the menu item **Edit** (or pressing **Tab**), clicking on **Indent Block**, then changing the 4 to a 2, then clicking on the **OK** button.



I also use **F4** for a four space indent and **F6** for six spaces. For the reverse I use **F3** for four spaces left and **F5** for six spaces left. These are examples of fairly simple but useful macros that are easy to remember. Remember the **F1** key is reserved by Clarion as the Help key (just put the cursor on top of a Clarion key word and press F1 and the appropriate Help screen will appear).

Another common need is to add a spaced equal sign " = " after a variable prior to adding another variable for an equate. I use F11 for this; 11 looks almost like 1=1, so it is easy to remember. After starting the macro, press these keys

```
End
SpaceBar
=
SpaceBar
F11
```

I reserve **F12** for "on the fly" or temporary macros that will not be saved. This is for odd situations that will vary session to session. If you find you are using the same **F12** function several times, then that is when you should save it permanently.

I reserve **Ctrl-F?** for various Clarion code snippets and **Alt-F?** for my own code snippets. For example, I use **Ctrl-F2** for a Fetch statement:

```
SpaceBar
SpaceBar
Access:File.Fetch(FIL:CodeKey)
Ctrl-F2
```

Now it is an easy fill in the blank line of code by highlighting and replacing File and FIL:CodeKey. An alternative to this is to put the Filename (I always use the DOS 8.3 concept for my names) and FIL:CodeKey on a line then use the following for the macro code:

```
Home Access:
RightArrow
```

```

RightArrow
RightArrow
RightArrow
RightArrow
RightArrow
RightArrow
RightArrow
RightArrow
RightArrow
RightArrow
RightArrow
.Fetch( End )

```

then close with the shortcut key again. This illustrates a more complicated way of automating your code. It can be very useful at times.

Here is another example of a more complicated routine which may be helpful. An alternative entry method follows also. I use Alt-F2 to read my Control File:

```

Enter
UpArrow
! ROUTINE -- getZCompany
Enter
Enter
getZCompany
SpaceBar
SpaceBar
SpaceBar
SpaceBar
SpaceBar
ROUTINE
Enter
Enter
SET( ZCompany )
Enter
LOOP
Enter
SpaceBar
SpaceBar
NEXT( ZCompany )
Enter
IF ERRORCODE() THEN BREAK.
Enter
BREAK
Enter
LeftArrow
LeftArrow
END ! LOOP
Enter
Alt-F2

```

When you press **Alt-F2** the following text will be placed at your embed:

```
! ROUTINE -- getZCompany
getZCompany      ROUTINE
  SET( ZCompany )
  LOOP
    NEXT( ZCompany )
    IF ERRORCODE( ) THEN BREAK.
  BREAK
END ! LOOP
```

Perhaps a better way to do this would be to store your code snippets or any other text needed in a Macro directory then use the macro to click on the menu item **File** then **Open**. Next, navigate to a directory where your files are, then select the file, then **Ctrl-A** to select all and **Ctrl-C** to copy. Return to the Clarion Editor. Place the cursor at the proper position then use **Ctrl-V** to insert the text. This would take an extra keystroke but would allow easy editing of the text and be far simpler to implement.

I recommend you make a keyboard template or strip that can be taped above the **F?** keys with a note as what each **F?** key does. This will make it easier to remember and use the shortcut keys not used every day.

In the Clarion 5 User's Guide on page 393 about the Text Editor, Macros, you can find some brief written documentation. Remember that macros are not saved in versions prior to Clarion 5.5. In Clarion 5.5 you will be asked to save the macro when you leave Clarion so it will be available later. I don't know the limit on the macro but I tested it to over 500 characters and it was still going strong. This would allow a lot of code and documentation. The macros are stored in the Bin\C55EE.DAT configuration file that is not readily accessible for editing.

The macro utility is a very valuable and powerful tool in C55. Use and enjoy.

---

*[Tom Giles](#) is a self-taught, long time CPD 2.1 programmer who started with interpretive BASIC for his business programs. He now uses CW5/5.5 for all new projects. When not programming he is out skydiving or flying his homebuilt airplane. Isn't life grand?*

## Reader Comments

[Add a comment](#)

Copyright © 1999-2003 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.





[Topics](#) > [Misc.](#) > [XML](#)

## XML For Clarion Developers

by **David Harms**

Published 2003-05-16

I remember hearing about XML in the late 90s, and wondering to myself what all the fuss was about. Here was a definition for storing data in text format. Text format? I was accustomed to flat file databases, and just warming up to SQL. What did I care about some quirky, storage-inefficient way of representing information?

I should have paid closer attention. XML has become a huge factor in software development. It is the language of choice for communicating data between systems with different native data formats (and architectures), for storing application configuration data, and even for storing instances of classes, data and all, for later use. And of course there are many other possibilities. In this article I'll provide a bit of an overview of XML, and a brief look at the XML technologies likely to be of greatest use to Clarion developers.

### What is XML?

What is XML really? For starters, XML is a markup language. That means it is used to add structure to existing information. You're no doubt at least somewhat familiar with HTML, another markup language. In HTML, for instance, you use the `<p>` tag to indicate a paragraph, the `<font>` tag to specify a font, and so forth. A tag, in both HTML and XML, begins with `<` and ends with `>` (which means, among other things, that if you want to use those characters in an HTML or XML document you need to use a special character string, `&lt;` for `<` and `&gt;` for `>`).

The superficial similarity between XML and HTML is no accident – both languages are subsets of [SGML](#), the Standard Generalized Markup Language. SGML is an enormously flexible markup language, and that flexibility makes it

unwieldy. You don't hear a lot about people hand-coding SGML documents. HTML on the other hand is very easy to use, but highly restrictive - there are really only a small number of allowed tags. New tags only come into wide use when they're supported by the major web browsers.

XML is a subset of SGML that retains much of that language's flexibility, but with ease of use. In XML you can create your own tags, and to some extent your own document structure. You don't have to wait for Microsoft to adopt your tags - you just make what you need.

You might wonder what good a bunch of arbitrarily defined tags are. Naturally any software that reads XML will need to know what to do with those tags, and that means you either write the software yourself, or you supply information to interpret those tags to someone else's software. I'll say more about that a little later.

There are a few key differences between XML tags and HTML tags. Tag usage in HTML is poorly enforced. For instance, you would normally use the `<p>` tag to begin a paragraph, and the `</p>` tag to end that paragraph. But the closing tag is not generally required, so you can have a bunch of paragraphs with only the opening `<p>` tag and the document will still display correctly. In XML, all tags must be closed at some point. That can either mean a pair of tags, as in:

```
<myTag>some data</myTag>
```

or it can simply mean a single tag with a closing `/`:

```
<myTag/>
```

The `<myTag/>` form is typically used where the tag produces some data for inclusion when, say, the XML file is processed for display.

HTML is pretty lax about overlapping tags. You may see this in an HTML document:

```
<i>Some <b>text</i></b>
```

Your HTML editor may complain about the closing `</i>` tag coming after the opening `<b>` tag, but there's a good chance your browser will let it pass. In XML, however, overlapping tags are strictly verboten (although nested tags are obviously permitted).

## An example

So what does XML look like? Here's a portion of a document, adapted from the `people_attr.xml` example that comes with Clarion 6:

```
<?xml version="1.0" ?>
<dataroot>
  <row>
    <id>1</id>
    <firstname>Fred</firstname>
    <lastname>Flintstone</lastname>
    <gender>M</gender>
  </row>
  <row>
    <id>2</id>
    <firstname>Andrew</firstname>
    <lastname>Guidroz</lastname>
    <gender>M</gender>
  </row>
  <row>
    <id>4</id>
    <firstname>Gavin</firstname>
    <lastname>Holiday</lastname>
    <gender>M</gender>
  </row>
  <row>
    <id>5</id>
    <firstname>Dick</firstname>
    <lastname>Tailor</lastname>
    <gender>M</gender>
  </row>
  <row>
    <id>6</id>
    <firstname>David</firstname>
    <lastname>Bayliss</lastname>
    <gender>M</gender>
  </row>
  <row>
    <id>7</id>
    <firstname>Claudia</firstname>
    <lastname>Steinburger</lastname>
    <gender>F</gender>
  </row>
  <row>
    <id>8</id>
    <firstname>Icky</firstname>
    <lastname>Smallhammer</lastname>
    <gender>F</gender>
  </row>
  <row>
    <id>9</id>
    <firstname>Spot</firstname>
    <lastname>Weasel</lastname>
    <gender>M</gender>
  </row>
</dataroot>
```

I deliberately called that an adapted example: actually, the `people_attr.xml` looks more like this:

```
<?xml version="1.0" ?>
<dataroot>
  <row id="1"  firstname="Fred"  lastname="Flintstone"  gender="M"  />
  <row id="2"  firstname="Andrew"  lastname="Guidroz"  gender="M"  />
  <row id="4"  firstname="Gavin"  lastname="Holiday"  gender="M"  />
  <row id="5"  firstname="Dick"  lastname="Tailor"  gender="M"  />
  <row id="6"  firstname="David"  lastname="Bayliss"  gender="M"  />
  <row id="7"  firstname="Claudia"  lastname="Steinburger"  gender="F"  />
  <row id="8"  firstname="Icky"  lastname="Smallhammer"  gender="F"  />
  <row id="9"  firstname="Spot"  lastname="Weasel"  gender="M"  />
</dataroot>
```

I've shown both versions to demonstrate that there are a number of different ways to present your data in XML. The first example uses four child elements for each row element; the second uses just a single row element, each with four attributes. As you can probably guess, this is data you'd be likely to see in a browse.

Besides adhering to basic XML syntax, XML documents can also be constrained to a specific structure.

## Schemas and DTDs

If you want your XML document to use only specific tags, values, attributes and data types, you will need to provide an additional document (or section in the XML document) which defines that tag usage. There are a number of different ways of representing this data, but the two most common are Data Type Definitions (DTDs), and Schemas. DTDs are an older approach, and are written in SGML. Schemas are a more recent development, are written in XML, and add a few additional features.

Schemas and DTDs are well beyond the scope of this article, so I won't go into any details here. Whether or not you need a Schema or a DTD depends on the complexity of the specification, the rigor of your development process, and the likelihood of someone creating a document that doesn't fit the specification. For a very simple application you can probably get away without one, although that's not generally the best practice.

DTDs and Schemas are important not just as design specifications, but for making sure the XML document conforms to the specification. You enforce this conformity with a *validating parser*. Parser? What's a parser?

## The parser

Although many XML documents are human-readable, and although you can create many of the simpler XML documents with a text editor or some simple code, chances are you'll want to use an XML parser to read/write XML data. A parser will also check for syntax errors, and possibly for structural errors.

When a parser reads an XML document it looks for any errors in the basic XML syntax, such as the aforementioned overlapping tags. If there are no syntax errors, then the document is said to be *well-formed*.

Some parsers are also able to check the XML document for structural errors. In order for a document to have structural errors, it must, of course, have a structure. If your XML document specifies that it conforms to a DTD or Schema, your parser is a validating parser, and you have told it you want it to validate documents, the parser will compare the structure to the DTD or Schema and report any errors. This is particularly important if, for instance, you publish an XML format for your customers, or the general public, and have no real control over the software used to create the documents you receive.

Parsers also handle character translation issues, such as changing the reserved < and > characters in the document's data to &lt; and &gt; when writing to XML, and back to < and > when reading XML. If the reserved characters were not translated, then document data could be misinterpreted as part of the XML markup notation.

Keep in mind that just because a document is well-formed doesn't mean it's valid, although a valid document is, by definition, also well-formed.

## SAX or DOM

There are broadly two kinds of parsers in the XML world: SAX parsers and DOM parsers. The difference between the two lies in how they process XML documents.

A SAX parser processes each line (or more accurately, each element) as it is read, much like Clarion processes ASCII files. As each element is read, you, the programmer, decide what to do with that element by means of callback functions. That is, you write a function which the SAX parser calls internally as it retrieves each element. SAX parsers are fairly efficient in their use of memory, and are a good choice particularly when you have a large XML document that

won't fit in available memory. The downside is that because of their sequential processing you can't use them to randomly read XML data.

DOM parsers, on the other hand, read the complete XML file before presenting you with a tree-structured document based on the file. If the SAX parser is like an ASCII file, then DOM is like an SQL table, where you can retrieve whatever data you like when you like (except that the data must all fit in memory). DOM parsers are typically read/write, while SAX parsers are typically read-only. For most business uses, especially with XML files that will fit in available RAM, DOM is probably going to be a better fit. If the file is too big, then your application will have to use virtual memory which could result in some disk thrashing. And keep in mind that a processed XML file will take up more memory than the raw data – I've read of anywhere from three to ten times as much memory! Your mileage may vary.

## Getting started

Clarion 6 contains an implementation of the [CenterPoint XML](#) parser which provides DOM Level 1 and Level 2 interfaces, as well as some documentation and an example app. Some of the C6 classes make up the wrapper around the parser; others provide a translation level between Clarion's complex data structures (including files, groups, queues and views) and XML documents. Clarion Magazine will contain more articles about this XML implementation in the future. If you're not using Clarion 6 EA (or, by the time you may read this, C6 gold or later) you can still read and write XML documents. Jim Kane has written articles about both [SAX](#) and [DOM](#) parsers for Clarion Magazine. And [ThinkData](#) sells a wrapper for the Microsoft XML parser: XMLFuse and other products are listed in the [ClarionMag product directory](#).

## Summary

XML has taken the software world by storm, promising, and in many cases delivering, new levels of interoperability between disparate programs and systems. To get an idea of how broad XML's appeal is, visit XML.org's [Schema/DTD registry](#). Here you can find XML specifications for everything from customer validation, to cave surveys, to marine trading.

Once you have chosen (or created) an XML specification, you can begin creating XML documents, typically with an XML parser. And you'll almost certainly use a DOM or SAX parser to process the XML data at the other end, turning it back into something your application can use.

If you haven't already been asked to provide or read data in XML format, you probably will be soon. Happily there are already several XML implementations available for Clarion 5.5 and Clarion 6.

---

*[David Harms](#) is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995). His most recent book is [JSP, Servlets, and MySQL](#), published by HungryMinds Inc. (2001).*

## Reader Comments

[Add a comment](#)

**Carl Barnes provided some important clarifications to the...**  
**Nice article! Readers do need to realize that this XML...**  
**Thanks Dave I've been getting my head around xml for a...**

Copyright © 1999-2003 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.



## [Clarion Magazine](#)



[Topics](#) > [News](#) > [ClarionMag News](#)

### **Clarion News**

#### [EasyExcel Special Offer](#)

First twenty (20) customers who buy EasyExcel 2.00 (full version) will receive a free copy of Product Scope 32 PRO V4.5(a), Spreadsheet Version, a multi-purpose tool used to display, create, organize, and research profile exchanges. Some currently available profile exchanges are Clarion 3rd Party, Search Engine, Home Theater, Newsgroup and Email, and PRO Music USB. Profile Exchanges are collections of data with common themes, categories or interest; designed to help you find information on the Internet more quickly in an organized fashion, locating related data and products. Soon to be released - Product Scope 32 PRO, Version 5.0 will include many image enhancements (viewers, thumbnailer, image reports, support for 15 image formats and more than 39 file extensions), PDF creation, and more spreadsheet enhancements.

*Posted Friday, May 30, 2003*

#### [CapeSoft Draw 2 Goes Gold](#)

CapeSoft Draw Gold Release is now available. Draw is a graphics acceleration library that provides replacements for the Clarion drawing commands, as well as a large number of additional features, including: Fast and flicker free drawing and display; ; Buffered for near instantaneous refreshes; Complex drawing features, like shaded boxes and cylinders, and per pixel shading and effects.; Save as PNG or BMP; Import BMP files and use them as a "brush" for drawing - including the ability to specify an index transparency color for the bitmap; Vertical Text supported; 3D and 2D geometry functions; Large number of graphics functions for creating both simple and complex objects; Auto shaded boxes, cylinders and primitive shapes; Load Windows XP icons, including the alpha channel; Works on both windows and reports; High speed 256 color images for reports and sending over networks; Thick lines and styles with all shapes. New in Draw 2: layers; alpha channels for "soft" transparency; index transparency (single color transparency); layers can be any size, and moved to any position, they can share alpha channels and even be hidden.; pixels can be copied from any layer to any layer, and even individual RGB components can be

copied between layers; Improved documentation and example programs. The upgrade from previous version of CapeSoft Draw to version 2.00 is free. Draw 2.00 will be on sale at the beta price of \$59 until 30 June 2003, when the price will change to the full Gold price of \$99.

*Posted Friday, May 30, 2003*

### **[Free Parallel Port Template](#)**

Danie de Beer has made available a free template for communicating with parallel ports directly from Clarion applications.

*Posted Friday, May 30, 2003*

### **[EasyListPrint Review](#)**

A review of EasyListPrint is now available on ClarionShop.

*Posted Friday, May 30, 2003*

### **[SimTabTree Template Special](#)**

The SimTabTree template allows you to drop a tree control onto a window to enable the selection of tabs on that window by clicking on the tree. Simplifies the look of multiple tabs by allowing you to set the wizard option (and the no-sheet option) then select the particular tab through the tree control. The opening price is \$19 US up to the end of June 2003 whereafter the price will be \$29 US.

*Posted Friday, May 30, 2003*

### **[Clarion 6 Early Access Release 4-3](#)**

Clarion 6 Early Access release 4-3 is now available to EA program participants. This release requires a rebuilt of all DLLs/LIBs due to changes in data initialization and constructor calls. Many bug fixes and changes, including a removal of the limit on the number of concurrently running threads.

*Posted Friday, May 30, 2003*

### **[New Version Of ABCFree Templates And Tools](#)**

A new version of the ABC Free Templates and Tools is now available. Changes include: Added M. Veenstra's LoadSubKeyQ method to the registry class; Added "substring" and "optional upper" options to the BrowseFilterFieldRange ('Browse: Filter based on field range') template. Clarion 6 testing continues. To date the only change required has been to the BrowseFilterFieldRange template to workaroud a bug in C6EA4.2 which is fixed in the latest C6 internal builds.

*Posted Monday, May 26, 2003*

### **[EasyResizeAndSplit 1.08](#)**

EasyResizeAndSplit 1.08 adds a new MDI interface (for C55 only). A new demo

is available. Now it is possible to have both maximized and non-maximized MDI windows simultaneously. Inactive MDI windows have no system buttons, they appear when the window becomes active.

*Posted Monday, May 26, 2003*

### **[cpTracker 50% Off Sale](#)**

From now until Midnight, Saturday, May 31, 2003, you may get your copy of cpTracker for ONLY \$65.00. This order page will be removed after the expiration date.

*Posted Monday, May 26, 2003*

### **[Riebens Systems Ends Product Development](#)**

Riebens Systems will be stopping development of new third party products for the Clarion Environment. All current Riebens Systems product source code will be made available for purchase over the next few months, starting with HTML Designer source code sometime during this month. Persons wishing to purchasing the source code may use the code for their own development and customization of products, but may not re-sell the code or re-market the product under an new branding.

*Posted Monday, May 26, 2003*

### **[Clarion SQL Seminar](#)**

Due to the immense popularity of the previous seminars, Shawn Mason is staging two more for the month of July. (Boston and England)

*Posted Monday, May 26, 2003*

### **[ImageEx And SysTrack Special](#)**

Everyone who buys ImageEx until May 31 will receive a copy of SysTrack free of charge. This does also apply to the competitive upgrade purchase option which will let you upgrade to ImageEx from any other third party graphics and/or imaging toolkit (proof of purchase is required)

*Posted Monday, May 26, 2003*

### **[Gitano 50% Off Memorial Day Sale](#)**

Gitano Software is having a 50% off Memorial Day sale. Sale products include logos, splash screens, icons, images, and Accent.

*Posted Monday, May 26, 2003*

### **[Lindersoft WebStore Officially Launched](#)**

The new Lindersoft WebStore is officially launched. This online store provides easy and secure online ordering for Lindersoft products. Products are delivered

electronically, which means you'll be able to use the software minutes after your order. Buy SetupBuilder 4 Standard or Web Edition now and get SetupBuilder 5 Standard (\$249 value) or Professional (\$399 value) free.

*Posted Monday, May 26, 2003*

### **[Lindersoft Vacation Notice](#)**

The Lindersoft office will be closed May 24 through June 02, 2003 for vacation. The purchase web site remains open - if you purchase a new license you will receive your registration codes immediately. Please note that it will not be possible to send out registration codes for updates or answer customer service email during this time.

*Posted Monday, May 26, 2003*

### **[MySQL Usage Notes](#)**

A reminder about Vernon Jay Godwin's document on using MySQL with Clarion - lots of good information here.

*Posted Monday, May 26, 2003*

### **[PD Browse Button Lookup 60-01 Beta](#)**

PD Browse Button Lookup Version 60-01 Beta is now available, including an all source code class library. This new version adds functionality while reducing the amount of code generated to only a few lines. The new class library is designed for use with both ABC and Legacy applications. It has been extensively tested with applications using the previous DLL version and should require few if any modifications in existing applications. The basic features of the lookup remain in place: Fill as you type entry with a browse button; Multiple lookups into the same file without aliasing; Easy handling of primary ID fields; Optional entry without a matching record in the lookup file; Optional disabling of lookups when a record is being changed; Faster data entry. New features include: Optional adding of records without calling a browse or form; Easy resetting of window on either an event accepted or new selection; Changed entries generated an event accepted (even though the lookup field has the IMMEDIATE attribute set); More but easier embed code hooks; Parameter passing to lookup procedures (part of last DLL release). During the Beta period the price is \$129 (\$79 for upgrades from the prior DLL version).

*Posted Monday, May 26, 2003*

### **[EasyListPrint 1.04](#)**

New in EasyListPrint 1.04: Template driven support for IceTips Report Viewer (including the new example application); Fix for long fields in standard report.

*Posted Monday, May 26, 2003*

## **[CPCS For EA4-2](#)**

All CPCS files have been rebuilt with EA4-2, and are available from the CPCS web site.

*Posted Tuesday, May 20, 2003*

## **[Plugware Email Offline For 12 - 14 Days](#)**

Plugware email will be down for about 12 - 14 days due to an ISP change while switching to DSL from ISDN. andy@plugwaresolutions.com will be temporarily replaced with andy@metz.com, but other Plugware accounts will not be available during this period.

*Posted Tuesday, May 20, 2003*

## **[ImageEx 2.1](#)**

ImageEx 2.1 is now available. New features include: Read-support for multi-page images (TIFF, PCD); A new class for reading EXIF and/or IPTC metadata from images; New method to adjust hue, saturation and lightness; New methods to add solarize and posterize effects; Auto contrast (stretching) function; SplitBlur and Add Noise (color or mono) effects; New class for saving GIF images (with RLE encoding, LZW encoding will follow after the UniSys patent expired); PNG saver class now optionally saves the alpha channel. This update is free for registered users

*Posted Tuesday, May 20, 2003*

## **[EasyListPrint 1.03](#)**

New in EasyListPrint 1.03: Added Excel reports (beta version, does not support MS Excel 97); Support for third party previewers (TinTools using template, all others using source coding); Fully customizable page setup and progress windows; EasyListPrint progress procedure - procedure template; Print entire BrowseBox; Pprint more then one report type; Bug fixes.

*Posted Tuesday, May 20, 2003*

## **[ControlMonitor Free Templates](#)**

This free template monitors all the controls and data on a window and executes code whenever any of the data is changed. This way you only need to put your code at a single embed point instead of on each control.

*Posted Tuesday, May 20, 2003*

## **[xDataBackupManager Pro v1.6](#)**

This release of xDataBackupManager Pro is a template bugfix with Windows XP

support. Updated Demonstration program and install kits for C5 and C55 are now available.

*Posted Thursday, May 15, 2003*

### **[xTipOfDay v1.7](#)**

This release of xTipOfDay is a template bugfix with Windows XP support. Updated Demonstration program and install kits for C5 and C55 are now available.

*Posted Thursday, May 15, 2003*

### **[Data Modeler 6.0 Update](#)**

The Data Modeler 6.0 for Clarion 6 is near completion and will be shipped shortly for Beta testing. The new triggers and rules associated with DM and the DCT file were only completed in Clarion 6 EA 4-1, therefore the delay. Localizer for Clarion 6 is already available. A totally new Documentation Expert 4 will be available soon.

*Posted Thursday, May 15, 2003*

### **[Logos, Design Banners, Etc.](#)**

1st Logo Design delivers web graphics, including logos, design banners, animated GIFs, splash screen, CD covers, and more. Ready to go package deals are available, or you can request custom design work.

*Posted Thursday, May 15, 2003*

### **[Icetips May 2003 Newsletter](#)**

The May Icetips newsletter is now available. Please note also that in the next month or so, Icetips' website presence will change drastically. Icetips.com will be the commercial website, and Icetips.net will take over the news and information part.

*Posted Thursday, May 15, 2003*

### **[Replicate Version 1 Beta 15](#)**

CapeSoft has released Replicate Beta 15. Changes include: The csLogConnectionManager class replaces the csLogEmailManager class - this new class incorporates FTP as well as making it easy to add your own transport mechanism (and expand this class to incorporate other transport methods); LogFile requisition is streamlined and implemented as a standard (without requiring extra coding); Imported logfiles are removed after import; LogFiles can now be generated in pure XML format (this opens the door for interaction with ASPs); Some bug fixes and other minor features.

*Posted Thursday, May 15, 2003*



### [PDF-XChange Twain/WIA Update](#)

This release of PDF-XChange resolves some problems and tidies up the classes initially released.

*Posted Thursday, May 15, 2003*

### [gReg Update](#)

An update is available for gReg which fixes the registration error and compile error for stand alone single exe apps. Available for v 4.51 C55 D, H and C5, v4.5 C55H and C5. Use the same passwords as before.

*Posted Thursday, May 15, 2003*

### [Image-XChange SDK Demos/Docs](#)

Additional demo apps and expanded docs are now available for the Image-XChange SDK.

*Posted Thursday, May 08, 2003*

### [CWPlus Review At ClarionShop](#)

There is new review of CWPlus available on the ClarionShop reviews page. CWPlus is an integrated tool for Clarion for Windows IDE, which realizes analogues of the most important elements of Microsoft IntelliSense technology as applied to Clarion, including: Pop-up help about the identifier; Help about the prototype of edited procedure; The actual list of identifiers with an opportunity to choose the necessary identifier and to insert it into the text; embed source reformatting. Clarion 6 version is free to purchasers of the C5/C5.5 versions.

*Posted Thursday, May 08, 2003*

### [SetupBuilder 5 News](#)

A partial list of the new SetupBuilder 5.0 features is now available; the complete list will be published when the beta of SetupBuilder 5 is released. Included features: new visual IDE (written with Clarion 6); SetupScript, a complete "drag and drop" setup scripting language with a full set of looping and flow-control statements; Include Scripts action; Visual Debugger; New Compiler and Linker; New Installer Engine; New LSPack 5.0 Compression Library, optimized for speed and compression ratio (the file size of the native 32-bit decompression library is down to 10 KB); New LSPatch Delta Patch Library; Dynamic Web-based installations; Application self-repair; Multi-language support for up to 25 languages by translating custom installer messages and dialogs. SetupBuilder 5 is currently under development and is not available for sale yet. Buy SetupBuilder 4 Standard or Web Edition now and get SetupBuilder 5 Standard (\$249 value) or Professional (\$399 value) at no additional charge.



*Posted Thursday, May 08, 2003*

### **[Scan2PDF Update In PDF-Xchange SDK 2.5](#)**

Scan direct to PDF is now available in the latest PDF-XChange SDK (version 2.5) using both TWAIN and WIA. An updated class/template and demo application is included.

*Posted Thursday, May 08, 2003*

### **[ClarionNET 1.3 Server Deployment Manager](#)**

All licensed users of the ClarionNET Server Deployment Manager have been notified by email of the free maintenance release for ClarionNET Version 1.3. The ClarionNET Server Deployment Manager Version 1.3 works in conjunction with new internal processes in ClarionNET 1.3 to provide a more stable and robust deployment system.

*Posted Thursday, May 08, 2003*

### **[Clarion TXA & TXD Scanner FreeWare](#)**

The Clarion TXA and TXD Scanner was designed to give a visual presentation of your source code and data dictionary without the need to load the data into Clarion. Procedure information, templates, variables, used files, images and icons , global templates and data are all displayed when a TXA is scanned. Table information, relations, fields and keys are displayed when a TXD is scanned. Unused tables are marked when both the TXA and TXD are scanned. Besides displaying this information on your screen you can copy fields and tables from the scanner into your Clarion dictionary. Also you can create program and dictionary documentation in HTML. The HTML colors are configurable as well as the specific data you wish to export. If you have the Microsoft CHM compiler installed then AppScanner can also generate a project for your CHM compiler and compile the HTML files to CHM giving you easy to distribute database and program layouts.

*Posted Thursday, May 08, 2003*

### **[cpTracker Lite Released](#)**

cpTracker Lite is an affordable Task Management System, which includes just the Task Management portion of cpTracker 2003. The price is \$27.00.

*Posted Thursday, May 08, 2003*

### **[Serial Comm Lib 1.1 New URL](#)**

Andrey Krivoshein's Clarion serial communications library has a new home.

*Posted Thursday, May 08, 2003*

## **[Fenix Beta Program Filled](#)**

The intended number of participants of the Beta program for Fenix ASP.NET Generator has been reached which means that the program is now closed to new participants. The price for Fenix is now \$ 699 (~~€660~~) (this price will be maintained until three months after the Fenix Gold Release). All new orders will be placed in backorder and processed when the Gold Release (expected at the end of June 2003) of Fenix is released.

*Posted Thursday, May 08, 2003*

## **[Image-XChange Clarion Templates](#)**

The Image-XChange SDK Clarion templates and classes are now available. For a limited time you can purchase the toolkit for \$299. Image-XChange can view, read and write to the following formats: BMP, GIF, JPEG, JNG, PNG, TIFF (Multipage), AMF/EMF/WMF (reads Vector/Raster but writes Raster only), PCX, TGA, JBIG2, with more coming soon, including DICOM for medical imaging purposes. A fully functional evaluation version with no time limits (but which puts a watermark on all saved file pages) is available. There are numerous example apps (with more coming to demonstrate CPCS, RPM, FOMIN, TINTOOLS etc. compatibility). Image-XChange also includes a new template/class to facilitate storing images as BLOBs.

*Posted Thursday, May 08, 2003*

## **[UltraTree Platinum 8 C6 Support](#)**

UltraTree Platinum version 8 supports Clarion 6 EA4, as well as all patch levels of Clarion 5.5 and ClarionNET.

*Posted Thursday, May 08, 2003*

## **[Vote On ETC-4 Date](#)**

If you're planning, or even just thinking really hard, about coming to the next ETC in 2004, the organizers need your input. The conference is currently scheduled for the last week of May, 2004 but it could be moved to either of the next two weeks: June 2-5 or June 9-12. Cast your vote today!

*Posted Thursday, May 01, 2003*

## **[RPM & PNet for EA4](#)**

Downloads of RPM and PNet for EA4 are now available. This install of RPM for C6 does not include the new upgrade. Due to the large number of additions and changes in the EA4 release it appears there is a lot more work to do. On the plus side: the preview window source, as well as source for almost all other windows (all but 2) - including all progress windows - is now provided. Also,

unless the additions to C6 warrant a change, there will only be two DLLs instead of the previous five.

*Posted Thursday, May 01, 2003*

### **Clarion Training in South Africa**

Incasu (Pty) Ltd will start presenting Clarion-related courses from May 2003 onwards in the Pretoria-area. Call 012 644 0400 for more information.

*Posted Thursday, May 01, 2003*

Copyright © 1999-2003 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.



[Topics](#)

## **Write For Clarion Magazine**

Clarion Magazine is looking for writers! If you have something of interest to Clarion developers, consider writing an article. You don't need prior writing experience, just a willingness to learn and share your knowledge with others.

Here are some of the ways you can write for Clarion Magazine:

- Write a feature article. Features are usually around 2000 words in length, and can cover everything from programming techniques to database theory to how to market your services. Here are just a few of the topics Clarion Magazine is interested in – if you have other ideas, [let me know!](#)
  - new features in Clarion 6
  - XML
  - using ClarioNET
  - using callback functions
  - web application development
  - prototyping C++/VB functions in Clarion
  - extending the ABC classes
  - examples of using a particular Clarion language statement
  - SQL techniques
  - MS SQL tips
  - working with multiple browses on one screen
  - using transactions (tutorial)
  - inner workings of ABC classes
  - using the TPS ODBC driver
  - interfacing with Word, Excel
  - using the Windows API
  - success stories
  - marketing your services
  - real-world experience with third party products (not a review, per se)
  - e-commerce
  - template programming

- programming standards
  - tales from the shop
- **Contribute a Tip or Trick.** These can be just a few paragraphs in length. If you have some code you'd like to share, or a useful way of doing something, you can just send us what you have and we'll write it up for you.

## **Why should you write for Clarion Magazine?**

There are several reasons why it makes good sense to write for Clarion Magazine. You'll be helping other developers who are facing the same challenges you've faced. You'll probably learn a few things in the process of writing the article (there's no better way to learn than to try to teach). You'll also get a small measure of fame (or perhaps notoriety).

## **Show me the money**

Of course, if you write a feature article or a programming tip, we'll pay you. Clarion Magazine pays authors on a royalty basis, so the more people who read your article, the more money you make. You may not get rich, but you'll probably make more than you think. You can also apply your earnings (paid in USD) to your Clarion Magazine subscription. This is a great way for non-US developers to beat the high cost of the US dollar.

## **What do I do next?**

If you have an idea for an article, email [editor@clarionmag.com](mailto:editor@clarionmag.com). If your idea is for a feature, and you're new to technical writing or you haven't written for Clarion Magazine before, you may also want to complete and submit a brief point-form outline (but it isn't essential).

Once your idea is accepted, go ahead and write the article. All submitted articles are subject to editing, and you may receive the article back several times with requests for changes, and to verify that any editorial corrections don't alter the meaning of the article. You should also submit a biographical paragraph so that readers can learn a little bit about you.

If you have any questions or comments about writing for Clarion Magazine, [email me](#).

Dave Harms

## Editor/Publisher

Copyright © 1999-2003 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

## [Clarion Magazine](#)



[Topics](#) > [Tips/Techniques](#) > [Windows API](#)

## **Veronica's Short History Of The Windows Operating Systems (Part 1)**

**by Veronica Chapman**

Published 2003-05-22

Clarion applications run under Microsoft's various versions of the Windows Operating System. In consequence, even though there is little reference to Clarion in this short history of Windows, everything here is highly relevant to Clarion.

It is true that the Clarion environment tends to, or tries to, hide as much of Microsoft's OS as possible. This is good for beginners, but bad for the experienced programmer. Sooner or later beginners (if they don't become discouraged) gain a certain amount of experience and "want more." At this point the Clarion environment starts to work against them because of the shielding it tries to provide.

Nevertheless Clarion does provide a full-access gateway to any of Microsoft's operating systems, for those intrepid enough to tread the path.

This gateway is (generically) called the Windows Application Programming Interface, or Windows API for short. Clarion fully supports calls to Windows API functions and procedures. These enable you, as a Clarion Applications Programmer, to do anything that anyone else can do in any other Windows language.

The purpose of this article is to help you:

- Know the Windows Operating System ... so that you know what you are ultimately dealing with, and
- Know how to use Clarion to drive the Windows Operating System in the direction you want to go.

What I state in this article is either provable fact (derived from inspection of the components involved), or a best guess, based on 30+ years of programming experience. I have never worked for Microsoft, and never will. Consequently my guesses may not be entirely correct in the finer detail; I trust they will be correct in essence.

## **Know the Windows Operating System**

As everyone knows, Windows started as a Graphical Implementation of MS-DOS, and this first edition culminated in Windows 3.x. Being based on MS-DOS it was a 16-bit implementation. I don't propose to dwell on this version, suffice to say that Windows 9.x incorporated the 16-bit implementation. Yes, evidence of Windows' 16 bit heritage is still there even today. Try typing Start-Run-progman.exe!

The main thrust of Windows 9.x was, however, to provide a 32-bit implementation of a Graphical User Interface (GUI). But Microsoft was desperate to rid itself of 16-bit once and for all, and so the company developed Windows NT (New Technology), based, not on MS-DOS, but on writing the whole thing from scratch (originally in collaboration with IBM, as part of the OS/2 project). In doing so the Windows developers went heavily into security features, multiple user accounts, and password protections. This provided them with the basis to create the Windows NT Server.

Of course compatibility was paramount, and so the Windows API for NT needed to incorporate (as far as possible) the regular Windows API. And so it does.

The switch to an all-32 bit Windows product line was not immediate, however. Windows 98, which still had 16 bit code, fixed loads of bugs and other issues that the 32-bit side of Windows 95 had exposed. Windows SE, ME and CE continued to do this while expanding the facilities available (sometimes taking backward steps in the process).

Now Microsoft had *two* implementations of their operating system, with the major goal being just *one*, based on 32 bit New Technology.

Enter Windows 2000, and Windows XP. XP is really a minor upgrade to Windows 2000 – the two operating systems are also known as NT 5.1 and NT 5, respectively. XP adds a new user interface and improved disaster recovery, among other features.



Ok. That's the overall picture ... but what has *really* happened, and how does it affect me as a Clarion Programmer?

Bill Gates was once reportedly quoted as saying "There's nothing like a nice piece of code". If he said this, then I (for one) fully agree with him because in my opinion the Windows Operating System is *nothing* like a nice piece of code. It has more than a passing resemblance to a lot of parts hastily thrown together. The original implementation (Windows 3.x) had almost no embedded file handling at all. A programmer would have to revert to MS-DOS calls to actually perform any useful data processing. This smacks of a large academic influence – heavy on the floating point and trigonometric functions, and very light when it comes to doing anything other than endless academic number-crunching. Absolutely brilliant if you want to evaluate Pi to millions of decimal places, but useless if you want to maintain a database.

But it was pretty, and it did make a computer look very sexy.

You might think the preceding paragraph is irrelevant complaining. It isn't. The Windows API of today is a direct descendant of the one described. We have to live with the ramifications of it even today.

The only time that Microsoft started Windows development from scratch was the 32-bit GUI implementation – from which the 32-bit side of Windows 95 was created and from which Window NT was created.

Apart from that, a newly-released Windows Operating System consists of:

- The previous Windows version (basically a collection of DLLs and drivers)
- New functions added to existing .DLLs, and
- Amended functions updated/expanded/debugged in existing .DLLs, and
- New functions added into new .DLLs

In *essence*, therefore, Windows 98 is just Windows 95 + a bit more, and so on. Again, *in essence*, this applies to Windows NT, Windows 2000 and Windows XP just as much as anything else.

The result is that a 'bog standard' application (that doesn't do anything fancy) will be forward compatible with any Windows version. Let's take an example: The most fundamental core of the Windows Operating System is the kernel, implemented as KERNEL.DLL under Win 3.x, and subsequently as

KERNEL32.DLL. Here's a picture of it's history, as taken from the Microsoft web site DLL Help Database, as shown in Figure 1:

- Clicking the 'more information' hyperlink on the bottommost entry (the earliest version) reveals KERNEL32.DLL as 'Visual Basic 5.0' and is dated '19<sup>th</sup> September, 1996'.
- Clicking the 'more information' hyperlink on the topmost entry (the latest version) reveals KERNEL32.DLL as 'Win XP SP (Service Pack) 1', and dates it as '29<sup>th</sup> August, 2002'. (We're into Service Packs on Win XP already!).

The screenshot shows the 'DLL Help Database' search interface. The search criteria are set to 'By File Only' and the file name is 'kernel32.dll'. The language is set to 'English'. Below the search form, it indicates 'Displaying Page 1 of 1.' and shows a table of search results for 'kernel32.dll'.

File Name	Version	More Information	Description
kernel32.dll	5.1.2600.1106	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	5.1.2600.0	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	5.0.2195.5400	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	5.0.2195.2778	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	5.0.2195.1600	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	5.0.2191.1	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	4.90.0.3000	<a href="#">More Information</a>	Win32 Kernel core component
kernel32.dll	4.3.0.1212	<a href="#">More Information</a>	Win32 Kernel core component
kernel32.dll	4.10.0.2222	<a href="#">More Information</a>	Win32 Kernel core component
kernel32.dll	4.10.0.1998	<a href="#">More Information</a>	Win32 Kernel core component
kernel32.dll	4.0.1381.4	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	4.0.1381.300	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	4.0.1381.3	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	4.0.1381.178	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	4.0.1381.133	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	4.0.1380.1	<a href="#">More Information</a>	Windows NT BASE API Client DLL
kernel32.dll	4.0.0.950	<a href="#">More Information</a>	Win32 Kernel core component
kernel32.dll	4.0.0.1111	<a href="#">More Information</a>	Win32 Kernel core component
kernel32.dll	3.51.1057.6	<a href="#">More Information</a>	Windows NT BASE API Client DLL

**Figure 1. The Windows DLL Help Database**

It is a similar story with, for example, USER.DLL ... which became USER32.DLL as the 32-bit implementation, also GDI.DLL which became GDI32.DLL, etc.

There hasn't really been a *new* Windows Operating System since Windows NT, and even NT retained Win 3.x compatibility ... because (at the time) it *had to*.

All that has happened since is that the original has been added to/updated and had bugs removed, etc. Most (if not all, really) of the facilities in common use today were built into the original 32-bit implementation. What has happened over the years is that additional lower level functions have been added to enable applications to get to previously hidden entities, and higher level functions have also been added to create re-usable objects, such as the Internet Explorer and its core component, called the WebBrowser.

The AOL Browser, for example, merely creates instances of the WebBrowser as it's "children" when showing multiple WebPages. The Internet Explorer creates a single instance of the WebBrowser, but spins off another instance of itself when showing multiple web pages.

Other browsers often use the 'WebBrowser' as a core-component. This is what [CapeSoft's File Explorer](#) does when you ask it to view an HTML page. Furthermore whenever you run a .CHM (Windows Help) file you will bring up the WebBrowser as a core component. Even those browsers that don't specifically use WebBrowser (e.g. Mozilla, Netscape) still use the same myriad of lower-level Windows API Functions that the WebBrowser collects together so conveniently (e.g. Dynamic HTML, Internet Session Handling, Internet Protocol Handling, Windows Sockets, etc.).

Like Mozilla, the WebBrowser/Internet Explorer (and all the rest) are just shells over the Windows Operating System. Consequently whichever Web Browser you choose often makes very little overall difference - if any at all. It might just be that you prefer the layout, and/or immediate facilities afforded by, a particular browser.

The point is that, because the foregoing is true, the Internet Explorer, or Mozilla, Opera, etc. could be written in the Clarion language just as easily (well, almost) as any other language. All the facilities are there. CapeSoft have done it.

## **Using Clarion to drive the Windows OS**

On the (possibly arrogant) assumption that no-one would bother to write a 16-bit application these days, the following discussion is for 32-bit implementations only.

To call the Windows API with Clarion you need to know a few Clarion-specific basics. The first of these is that calling the Windows API functions the same way you call regular Clarion function calls simply won't do the job. You need to adjust the function declaration in the Clarion MAP area for the call to work.

This adjustment consists, at a minimum, of remembering to add the attributes `PASCAL` and `RAW` to all Windows API function/procedure calls. For example the API function `GetWindowText` will return the caption from any window currently in existence.

Microsoft's documentation shows the `GetWindowText` prototype thus:

```
int GetWindowText(
HWND hWnd,          // handle of window or control with text
LPTSTR lpString,   // address of buffer for text
int nMaxCount      // maximum number of characters to copy
);
```

C Programmers have to make sense of these hieroglyphics. In Clarion, this translates to something more familiar, if not much more legible:

```
GetWindowText(LONG hWnd, |      ! Window Handle
 *CSTRING WindowCaptionBuffer, |
 LONG sizeCount), LONG, PASCAL, RAW, NAME( 'GetWindowTextA' )
```

First of all the `int` that prefaces the `GetWindowText` in the Microsoft C-style definition translates to the `LONG` return type in the Clarion version. This is then followed up by the obligatory `PASCAL` attribute, to tell Clarion to use the `PASCAL` parameter-passing convention (parameters go on the stack, not in the registers, and furthermore the sequence is from left to right), and `RAW` to tell Clarion that any strings are passed by address alone (Clarion normally also passes the length of the string, something totally unknown and unacceptable to the Windows API).

Then follows the re-naming directive `NAME( 'GetWindowTextA' )`. This is necessary because there are two versions of `GetWindowText` in the 32 bit Windows API – there is a regular ANSI version (`GetWindowTextA`), and a Unicode (or "wide") version called `GetWindowTextW`. This re-naming is generally only necessary for API functions where text strings are involved.

To determine whether you need the `NAME` attribute or not, you can either use `LIBMAKER` to list the functions in a DLL, or you can leave the re-naming off and try to compile/link. If it links – all well and good. If you get a linker error (unresolved), try the `NAME` attribute with an `A` at the end of the regular name.

These methods generally succeed, however there are variances between 16-bit compilations and 32-bit. My suggestion (if you don't already) is to always compile your Clarion applications in 32-bit.

There are still times PASCAL, RAW, and NAME won't do the trick – where you are using some fairly exotic Window API function that the Clarion linker is unable to recognize. If you encounter this situation you would need use LIBMAKER to create a .LIB from the appropriate Windows DLL (I'll have more to say about LIBMAKER at the end of Part 2)

But I was talking about GetWindowText. I've described the return part of the prototype – LONG, PASCAL, RAW, NAME( 'GetWindowTextA' ). You'll also need to deal with the three parameters: HWND hWnd, LPTSTR lpString, and int nMaxCount. These bring up a somewhat lengthy discussion, which I'll get into next week.

---

*Veronica Chapman earned a B.Tech in Electronics & Electrical Engineering from Brunel University in 1968, and subsequently embarked on a career as a Programmer/Analyst, first writing code at machine level, and shortly thereafter working with real time systems and communications. By the mid '70s she was using languages such as COBOL and FORTRAN. Never a fan of BASIC or the C language, she discovered Clarion in the mid-90s and, ever since, has used it to create applications for the 16-bit (Win 3.x) and 32-bit Windows Platforms. An assembly code programmer from way back, Veronica discovered a cornucopia of very useful functions in the Windows API, and set about making these functions available to Clarion applications.*

## Reader Comments

[Add a comment](#)

**Can't wait for part 2!**

**Thank you!**

Copyright © 1999-2003 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.



[Topics](#) > [Templates](#) > [Templates, writing](#)

## Finding Source With Enhanced Templates

by **Steffen Rasmussen**

Published 2003-05-23

One of the really great features of Clarion is the templates, and over the years the number of templates I use in my applications just keep growing. Although templates are wonderful they can also be the source of great frustration. I don't know how many times just a small change in the dictionary creates havoc when I try to compile the program associated with that dictionary, because the template is unable to handle the dictionary change. Not to mention the frustration of trying to find the offender, and before you know it the day has passed and what have you done? – all because of this minor change in the dictionary.

How hard can it be to track down dictionary-related template problems? Unfortunately, a lot of times it is quite hard, and when you realize that the culprit is in a template the big question is which one is causing the problem? That in itself can be a major challenge. For instance, I recently changed a field in the dictionary and when I compiled the program it just stopped in the middle and threw 15 errors at me. I had an idea where to find all of them except for one. It took me forever to realize that the source that was causing the compiler error came from the **SaveButton** control template where the **Field Priming on Insert** was set for this field.

### Template Enhancing

My frustration made me think that at least I could improve my own templates in order to minimize this problem in the future. The solution I came up with isn't exactly rocket science, but then again, the simpler the better.

In a nutshell my solution is to have each template place some code in the source based on the programmers input. Even a very simple template can scatter source code all over the place; you can document all this code just by adding a comment

to every single template source line. The only question is what this comment should contain?

In order to help the programmer, the template comment should contain at least the name of the template. Another nice thing to know is what control this template is used with (if any) and where in the template properties this particular code segment is set.. Of course as a template writer you would also want to include you're basic level of code documentation.

Let me give you an example. Currently I am working on an Edit-In-Place template called "EIP" which I want to document. So the first thing to include in the code comment is:

```
#<!EIP
```

Here the #<! string is used for aligning the code comments evenly in the source based on the template command #COMMENT which sets a column start number for these comments. The EIP string is the name of the template.

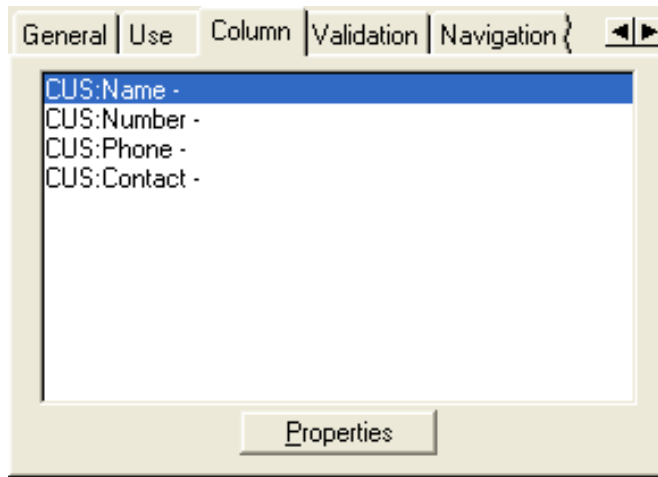
The next thing I want to show is the control in which this template is used. For this I use the template statement %Control. Using template statements to generate the information needed is essential for two reasons. The first reason is that the information changes depending where the template is used. The second reason is that it is much easier to copy/paste symbols than it is to copy/past/change comments that don't make use of template symbols.

My EIP template includes a list where the programmer can specify multiple sets of values for the underlying prompts. Instead of forcing the programmer to traverse through every single value to find a particularly prompt value I include a description that distinguishes each line in the list. For example the following template code creates a list of controls:

```
#BUTTON('EIP Properties'),FROM(%ControlField,%ControlField),INLINE
```

In this list the second instance of %ControlField is the description, which distinguishes each line, as shown in Figure 1.





**Figure 1. The %ControlField List**

This template also uses a series of classes the programmer can select, e.g. `EditCheckClass`; depending on which class has been selected, specific code for this class is added to the source. That's another bit of information I want to add to the comment. The last thing to include is the code documentation, so I now end up with the following comment line:

```
#<!EIP-%Control-%ControlField-%EIPClass-Rewrite current value
```

and in the actual generated source the comment looks like Figure 2.

```
IF Access:InvDet.Fetch(IND:InvDetIdK)-Level:Benign !EIP - ?List - IND:Print - EditCheckClass - Get Record
IND:Print=CHOOSE(IND:Print=1,0,1) !EIP - ?List - EditCheckClass - IND:Print - Change Check box value
IF Access:InvDet.Update()-Level:Benign. !EIP - ?List - IND:Print - EditCheckClass - Rewrite current value
DetailList.ResetQueue(Reset:Queue) !EIP - ?List - IND:Print - EditCheckClass - After update Refresh Q
DetailList.PostNewSelection !EIP - ?List - IND:Print - EditCheckClass - Refresh window
END !EIP - ?List - IND:Print - EditCheckClass - End if HouseDownField
```

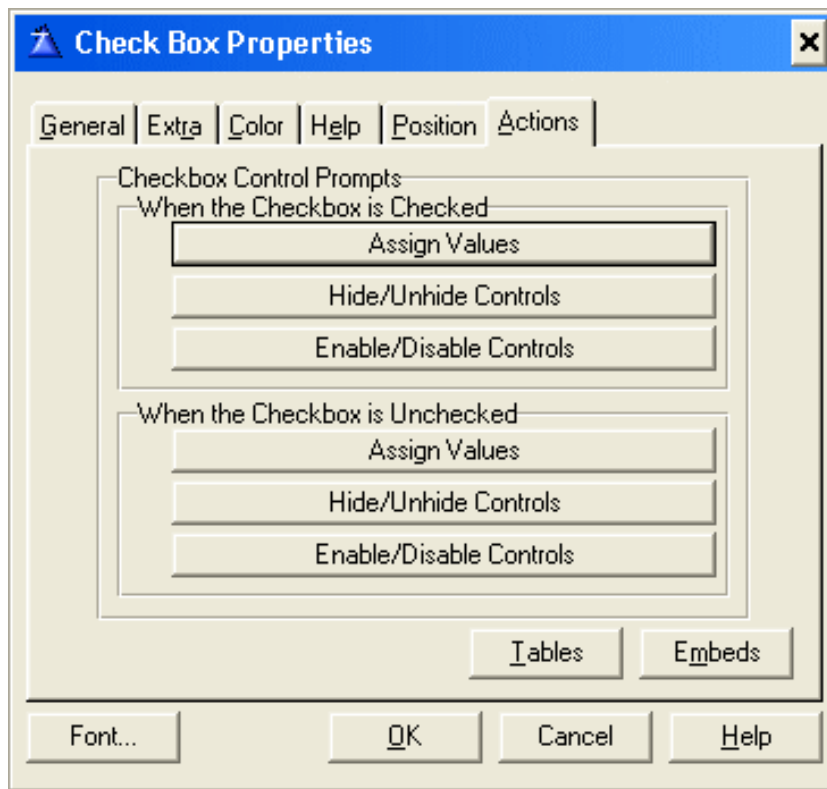
**Figure 2. Generated source ([click here](#) for the full sized image)**

As you can see it is now possible to quite accurately determine the template used (EIP), in what control (?List), the name of the column, the class selected and what this code line is doing.

## Every click counts

It's one thing to find the template causing a compile error, but it can also be quite annoying to have to go through the template menus to figure out what prompts has been set. Take the check box template, for example. On the Action tab there are six buttons (see Figure 3):

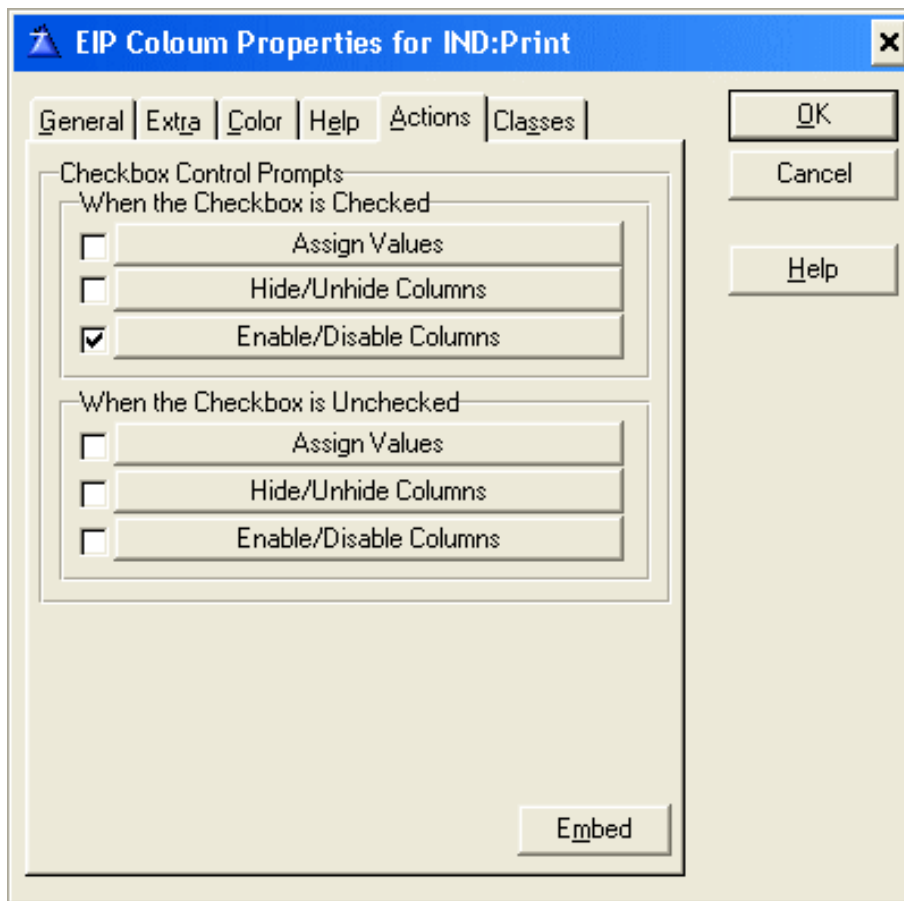




**Figure 3. Original check box template Action tab**

So if you want to determine what prompts in the template have been set you will have to open and close all of these buttons. That's at least 12 clicks if they are *not* populated. Now if there was a visual clue that indicated whether or not the underlying prompts are populated I could end up saving a lot of time.

To add that visual clue I implement an extra check box for every template button (see Figure 4). Then I check it on if there are underlying prompts that are set, and if there is none I check it off.



**Figure 4. New check box template Action tab**

These check boxes are not supposed to be set by the programmer manually; instead the template itself automatically assigns a value when appropriate. So the overall structure of each template button is as follows:

```
#BUTTON('Assign Values'),MULTI(%EIPMultiCheckedAssignValues, |
  %EIPCheckedVariableToAssign&amp;' = '&%EIPCheckedValueToAssign)
#DISPLAY('Values to assign when the check box is checked')
#DISPLAY('')
#PROMPT('Variable to Assign:',FIELD),%EIPCheckedVariableToAssign
#PROMPT('Value to Assign:',@s20),%EIPCheckedValueToAssign
#ENDBUTTON
```

Apart from each button I also have to include the check boxes, as you can see above in Figure 4. The code for the check box looks like this:

```
#PROMPT('',CHECK),%EIPMultiCheckedAssignValuesStatus
```

Notice the #PROMPT name which is an empty string. This is because I don't need any text for the check boxes.

The next step is to program the detection of the underlying prompts being used to determine if the check box should be on or off. In the template language you

cannot directly program an action into the template IDE. Instead you can call a group in which you can assign the appropriate values with the template command WHENACCEPTED.

The template #GROUP command is basically equivalent to the standard Clarion Procedure Routines. In this case I am just going to use it in its simplest form with out parameter passing. So how do I determine if the underlying prompts are populated? There are basically two things I have to take into consideration. The first is that the button can open a window containing a series of prompts. The second is that the button can open a window containing a list with multiple instances of the same prompts.

In the first case with the new window just containing prompts I have to determine if any of the prompts are populated. What would be really easy is if one of the prompts is required. In that case I can just check that one prompt; otherwise I would have to check them all. In that case the code would look something like this:

```
#GROUP(%GroupEIPPromptUsed)
#IF(%Prompt1)
    #SET(%EIPMultiCheckedAssignValuesStatus,1)
#ELSIF(Prompt2)
    #SET(%EIPMultiCheckedAssignValuesStatus,1)
#ELSIF(Prompt3)
    #SET(%EIPMultiCheckedAssignValuesStatus,1)
#ELSE
    #SET(%EIPMultiCheckedAssignValuesStatus,0)
#ENDIF
```

The second thing I have to take into consideration is if the #BUTTON calls a window with a list containing multiple underlying prompts. Initially this looks more complicated then the case described before. But since it is a list I know that if it contains anything, then row one has to be populated. If row one is empty then there is nothing in the list. So in this case the code would look something like:

```
#GROUP(%GroupEIPPromptUsed)
#SELECT(%EIPMultiCheckedAssignValues,1)
#IF(%EIPCheckedVariableToAssign)
    #SET(%EIPMultiCheckedAssignValuesStatus,1)
#ELSE
    #SET(%EIPMultiCheckedAssignValuesStatus,0)
#ENDIF
```

Of course the complexity can be greatly increased if you have a mix of the two instances just described, where you have one template window calling another,

which then again calls yet another etc.

Now that the #GROUP has been created I just have to call it:

```
WHENACCEPTED(%GroupEIPPromptUsed ( ))
```

Notice the ( ) after the group name; this tells the template that you are calling a group. The next thing you have to do is determine which prompt has to include the WHENACCEPTED command to call the group. One place where it always has to be is after the check box prompt, so if the programmer decides to check or uncheck it, a call to the group will be executed and the correct value will be assigned regardless of what the programmer may want it to be. Another place where you will have to place the #GROUP call is on one of the template prompts containing the REQ attribute within the window being called. If you have no way of knowing which one of all the prompts will be set, you will have to make a #GROUP call after every one of them since there is no way to determine if the template windows **OK** or **Cancel** buttons has been pressed.

In the situation where the template button calls a window with a list you have no way of knowing if the contents of the list has been deleted. Therefore I usually also make a #GROUP call when the prompt button is selected. If it is opened and the list is empty the #GROUP will be called and update the template check box, so I don't open it again unless I want to populate it. For this particularly instance I am still looking for a better solution.

With this article I have included a small extension template sniped from my EIP template showing a working example of what I have just described.

[Download the source](#)

---

*[Steffen S. Rasmussen](#) has graduated in Computer Science from Copenhagen Business College. Since then he has worked as a programmer, system technician and network administrator, and is currently IT manager.*

*Clarion is a quite a new language to Steffen since his only been working with it since January 2000. But what better way to learn it than by trying to teach others! Steffen has also set up a [web site](#) to collect as many examples of different user interfaces as possible to inspire Clarion developers.*

## Reader Comments

[Add a comment](#)

**Good idea. However, there is only one small problem with...  
where I come from we like to leave the templates alone and...**

**Alptekin, I totally agree with you, this is not a...**

**Quick template for commenting other templates I too am...**

**Alptekin makes a great point about not modifying the...**

**Casey, Why didn't I think of that . Excellent idea.**

Copyright © 1999-2003 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.

Reborn Free

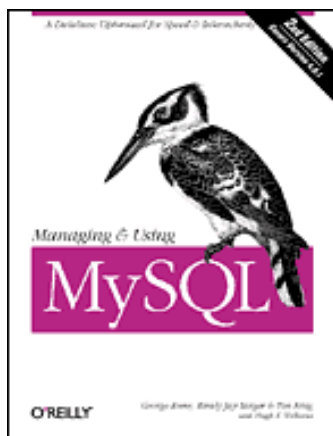
CLARION  
*online*

[Topics](#) > [Databases](#) > [MySQL](#)

## Book Review: Managing & Using MySQL

by **David Harms**

Published 2003-05-23



### [Managing & Using MySQL \(2nd Edition\)](#)

By George Reese, Randy Jay Yarger, Tim King  
With Hugh E. Williams

published by O'Reilly, April 2002

ISBN: 0-596-00211-4

424 pages, \$39.95 US, \$61.95 CA, £28.50 UK

Managing & Using MySQL is a well-structured, well-written book that serves as a solid introduction to both MySQL and SQL databases in general. It also provides an introduction to some of the numerous development environments which work with MySQL, such as PHP, Perl, and Java. Curiously absent is any significant information about Windows development and ODBC, but Clarion developers, at least those new to SQL, will still find much here of interest, as will Clarion developers who dabble in some of the other languages the book discusses.

Managing & Using MySQL is divided into four parts. The first part covers MySQL and SQL basics, installation, and database administration. As with the rest of the book, this part discusses MySQL really only in reference to itself. Absent (aside from a few brief mentions in the opening chapter) are any comparisons between MySQL and, say, Oracle, MSSQL, or PostgreSQL. The authors are also relatively uncritical of MySQL's feature set as compared to more "mainstream" SQL databases. For instance, foreign keys are dismissed

with the statement “Applications themselves should generally worry about foreign key integrity.” Happily, that’s something Clarion is quite able to do; still, if you’re looking for a book to help you choose between databases, this isn’t the one. What you will get (at least to begin with) is a very solid introduction to SQL basics and good instructions on MySQL installation and setup.

Part II has chapters on performance tuning, security, and database design. I particularly appreciated the discussion of the `EXPLAIN SELECT` command, which you can use to find out what steps the server is actually taking when it processes a `SELECT`. This information can be critical to improving performance. I also was very impressed with the chapter on relational database design; I found it a concise, readable introduction to the topic.

Part II ends at page 134 of some 400 pages. Part III, which is about as long as parts I and II combined, is primarily made up of chapters introducing MySQL development with Perl, Python, PHP, C, and Java. This is in keeping with the introductory nature of the book, and some chapters will be of interest to some Clarion developers, but I couldn’t help but feel a bit let down; I really wanted this book to continue on to some advanced topics. On to Part IV.

The final part (a.k.a. the last third, by volume) of this book is reference material: SQL syntax; MySQL data types; operators and functions; PHP API reference, C reference, and the Python DB-API. There’s some value added here, but I think this part of the book is a bit long for what is still mainly a regurgitation of information available online.

Note: This book covers MySQL 4.01; version 4.1, which adds some significant features such as subselects, is now in alpha release.

If I sound a bit critical of this book, it’s only because it started so well, and I came to expect so much. Although half the book may not be of interest to most Clarion developers, I think it’s still worth the price of admission, particularly for those just starting out with SQL and MySQL.

---

*[David Harms](#) is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995). His most recent book is *JSP, Servlets, and MySQL*, published by HungryMinds Inc. (2001).*

## Reader Comments

[Add a comment](#)

Copyright © 1999-2003 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.



online sales and delivery  
for your applications & tools

Developer **PLUS**

[Topics](#) > [Tips/Techniques](#) > [Windows API](#)

## **Veronica's Short History Of The Windows Operating Systems (Part 2)**

**by Veronica Chapman**

Published 2003-05-28

[Last week](#) I provided an overview of the Windows API, and began a discussion of the `GetWindowText` API call. I described the translation of the last part of the prototype – `LONG, PASCAL, RAW, NAME( 'GetWindowTextA' )`. This week I'll show how the parameter part of the prototype is translated. I'll also describe how to use `GetWindowText` along with the `GetClassName` function to provide an enumeration of all top-level windows, together with their children. And finally I'll provide a modified `LIBMAKER` utility that fixes a small but potentially troubling problem with `LIBs` created from Windows `DLLs`.

### **Translating parameters**

By way of review, here is the original C style declaration for the `GetWindowText` API call:

```
int GetWindowText( HWND hWnd, // handle of window or control
    LPTSTR lpString, // address of buffer for text
    int nMaxCount // maximum number of characters to copy
);
```

And here is the Clarion equivalent:

```
GetWindowText(LONG hWnd, | ! Window Handle
    *CSTRING WindowCaptionBuffer, |
    LONG sizeCount), LONG, PASCAL, RAW, NAME( 'GetWindowTextA' )
```

The three parameters of `GetWindowText` translate from the three in the C-style definition under the following guidelines for handles and zero-terminated (or `ASCIIz`) strings: (Note: The translation of `GetClassName` is almost identical).

## Handles

The Windows Operating System uses a myriad of "handles." There are Window Handles, Region Handles, Device Context Handles, Event Handles, Object Handles, Process Handles, Thread Handles ... you name it ... there is a handle for it. A handle is really nothing more than a unique identifier for something in memory.

Microsoft documentation stresses that a handle should *not* be considered to be any kind of index or pointer. That is, you can't *assume* that the handle is the memory address of a structure; it may be, or it may be an index into a table of pointers, or something else.

In the 32-bit implementation these handles tend to appear as the type HANDLE (or one of its variations, such as HWND, HPEN) in C-style definitions, and in 32 bit Windows this generally means a 32-bit word, which is a LONG in Clarion.

**Note:** The most common view of any handle is as a hexadecimal value (rather than, for example, a decimal value). All standard tools present handles in hex.

However, there remains a compatibility throwback to previous Windows Operating Systems, as I indicated [last week](#). Although handles are now 32 bits wide, *window handles* specifically appear to still be 16-bit in terms of their *range* of values (i.e. 0000h to FFFFh). It is possible to have a 32 bit window handle that is higher than FFFFh, but the first 16 bits (as far as I know) are always unique, and apparently sufficient to identify the window. I assume this was necessary for reasons of compatibility when Windows 95 started to replaced Win 3.x.

Under Win16 handles are defined as 16-bit (SHORT) values. When developing 16 bit apps I generally used USHORT because handles are not positive or negative values – to all intents and purposes they are just unique 'bit patterns' to uniquely define an entity.

All new handles introduced with Win32 are (of course) defined as 32-bit (LONG). Even though they remain just bit patterns, I don't use ULONG because I have had situations where testing Zero/Non-Zero has failed to run correctly under the ULONG definition, whereas simply changing to LONG and recompiling works fine. I put this down to a Clarion compiler bug. (Editor's note: ULONGs are handled as DECIMALs when the compiler does math, and a ULONG

comparison bug was reported and fixed in C6 in January.)

When using your Clarion Compiler to compile 32-Bit, simply use LONGs for all handles – even if only the first 16 bits are used for a window handle, it won't matter. When the compiler PUSHes and POPs a function's parameters onto, and off of, the stack, 32 bit machines will always use a 32 bit word. If the API function happens to use less than the entire width of the 32 bit machine word it will not in any way affect the validity and operation of your API call.

Alternatively, if you wish to be able to compile either 16 or 32 bit, you can use the Clarion compiler to make the necessary adjustments for you by means of conditional compilation. This is the technique employed by the various C compilers and Visual BASIC. The example that accompanies this article demonstrates the mechanism, which employs Clarion's built-in `_width32_` flag to do the trick (Editor's note: `_width32_` is deprecated in C6, which only supports 32 bit applications).

## ASCIIz strings

The ASCIIz string (zero- or null-terminated string) is native to the Windows API. The equivalent in Clarion is not a STRING but a CSTRING. Furthermore CSTRINGS are always passed to the Windows API *by address*. As I'm sure you know, in Clarion you signify a parameter passed by address by prefixing the declaration with a `*`; that's how `*CSTRING` becomes the second parameter.

The third parameter to `GetWindowText` is simply an integer value which tells the Windows API the maximum size of the buffer you have supplied – or you can use it to impose a lower limit on the characters copied. Consequently a LONG (LONGs for everything, remember) is the appropriate definition here.

These are the general rules of translation. I've translated hundreds of Windows API calls, and anyone interested is welcome to [contact me](#) if they have a translation problem, or if they want a copy of my ever-expanding set of API function declarations.

## Using GetWindowText

The next question (and the last of this article) is how/when/where would `GetWindowText` be used? You can always get your current Clarion window handle by using the Clarion window property `{PROP:Handle}`. However, you can also get your Clarion window caption/text by using the Clarion window property `{PROP:Text}`, so `GetWindowText` doesn't really do a whole lot in

relation to your current Clarion Window.

However, `GetWindowText` will obtain the caption for *any* open window anywhere in the system – all you need is the appropriate window handle and a `CSTRING` buffer in which to place it.

There are several ways to get the window handle. If, for example, you want to investigate all the top level windows open at any one time, you can use the API function `EnumWindows`. This calls an *enumerations processor* function you define. Your function is called by `EnumWindows` once for each top level window open at the time. The information passed to your enumerations processor (otherwise known as a callback function) includes the necessary handle.

At the end of this article you can find a downloadable zip with a demo app that shows how to use `GetWindowText` to get the caption of all top level windows currently open on your computer.

Here's the Clarion prototype for the `EnumWindows`, in C notation:

```
BOOL CALLBACK EnumWindowsProc( HWND hwnd, LPARAM lParam );
```

And this is the equivalent Clarion declaration, using the function name `TopLevelEnumProcessor` (the name is arbitrary):

```
TopLevelEnumProcessor FUNCTION(LONG FnhWnd, |  
LONG FnlParam), BOOL, PASCAL
```

You'll notice this function returns a `BOOLEAN` result; there is no native Boolean data type in Clarion, although most developers faced with a true/false result would probably use an 8 bit `BYTE` variable. Actually the `BOOL` keyword is allowed in Clarion – it's a synonym for `SIGNED`, which in 16 bit it is `EQUATED` to a `SHORT`, and in 32 bit to a `LONG`. Back to `LONG` again!

You have a number of options for dealing with the return value in any procedure. If you don't need it, you can add the `PROC` attribute, telling the Clarion compiler to throw away the return value without generating any compilation warnings. Personally I don't view this as a very good practice, so I always define a variable, `WBR LONG` (or `BOOL`, if you're in the Application Generator) for Windows Boolean Return in Global Data. This helps to remind me that I'm dealing with a return value – even if I often throw it away.

As I said, the enumeration function will call `GetWindowText`. Besides

Boolean results, most Windows API functions return either a SHORT (16 bit) or LONG (32 bit) value. This value is quite often an error indicator. In the case of `GetWindowText` the value returned is the number of characters copied from the caption of the window into the buffer you provided. This function won't overrun your buffer because you will have specified the maximum length in the third parameter. The easiest method of doing this is to specify the integer value for the third parameter as the expression  $(\text{SIZE}(\text{ASCIIZWindowCaptionBuffer}) - 1)$ . If you later adjust the size of your `ASCIIZWindowCaptionBuffer` CSTRING, the Clarion compiler will take care of the corresponding adjustment to the third parameter automatically.

## Putting it all together

The following is one way to implement `EnumWindows` and `GetWindowText` to retrieve the handles and captions of all windows currently open on your computer. These instructions are for adding this capability to an APP; for a PRJ see the downloadable source.

In your application's Inside the **Global Map** embed, place the following:

```
module('win32.lib')
  EnumWindows(TopLevelEnumProcessor |
    lpEnumFunc, LONG lParam), PASCAL, BOOL
  GetWindowText(LONG hWnd, *CSTRING WindowCaptionBuffer, |
    LONG sizeCount), LONG, PASCAL, RAW, NAME('GetWindowTextA')
end
```

In your application's **Global Data** section embed the following:

```
WBR BOOL
```

In **Global** or **Module Data** (depending on how widely you want the queue available):

```
TopLevelCaptions    QUEUE, PRE(TL)
WindowHandle          LONG
WindowCaption         CSTRING(101)
END
```

Now create the callback function. Create a new Source procedure called `TopLevelEnumProcessor`. Set the **Prototype** field to  $(\text{LONG FnhWnd}, \text{LONG FnlParam}), \text{BOOL}, \text{PASCAL}$  and the **Parameters** field to  $(\text{LONG FnhWnd}, \text{LONG FnlParam})$ . Check the **Declare Globally** checkbox.

The prototype for `TopLevelEnumProcessor` (and any callback you write) *must* have the `PASCAL` attribute. Since Windows API callbacks don't use strings, you don't need the `RAW` attribute, which is really fortunate because Clarion won't allow you to specify it anyway (you can't use `RAW` on a Clarion function – you can only use `RAW` on an *external* function, such as a Windows API Function).

Now embed the following in `TopLevelEnumProcessor`'s code section:

```
TL:WindowHandle = FnhWnd
  IF GetWindowText(TL:WindowHandle, |
      TL:WindowCaption, |
      (SIZE(TL:WindowCaption) - 1)) = 0
    TL:WindowCaption = '*Null*'
  END
  ADD(TopLevelCaptions)
  RETURN(True)      ! True = Continue Enumerating,
!                  False = Stop
```

Finally, carry out the enumeration by calling `EnumWindows`, passing the `TopLevelEnumProcessor` procedure as a parameter. You can do this, say, from a button's **Accepted** embed:

```
FREE(TopLevelCaptions)
WBR = EnumWindows(TopLevelEnumProcessor, |
0)! (This value becomes FnlParam)
```

With regard to reference material, the best I've come across is Microsoft's own. I have a `Win32API.HLP` file that acts as my "preliminary" bible, but the MS Online Library is the best source of information:

<http://msdn.microsoft.com/resources/libraries.asp>

I hope you have found this article informative. To accompany it I offer a working implementation of `GetWindowText` and `EnumWindows` (and a few other APIs as well) as a compilable Clarion project. This project contains a Windows Help File which explains the example in depth.

[Download the source code](#)

---

*Veronica Chapman earned a B.Tech in Electronics & Electrical Engineering from Brunel University in 1968, and subsequently embarked on a career as a Programmer/Analyst, first writing code at machine level, and shortly thereafter working with real time systems and communications. By the mid '70s she was using*

*languages such as COBOL and FORTRAN. Never a fan of BASIC or the C language, she discovered Clarion in the mid-90s and, ever since, has used it to create applications for the 16-bit (Win 3.x) and 32-bit Windows Platforms. An assembly code programmer from way back, Veronica discovered a cornucopia of very useful functions in the Windows API, and set about making these functions available to Clarion applications.*

## Reader Comments

[Add a comment](#)

**Very interesting Veronica! Thanks a ton.**

Copyright © 1999-2003 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.

[Topics](#) > [Browses](#) > [Browses, Using](#)

## A Tree In A Page Loaded Browse

by **Ronald van Raaphorst**

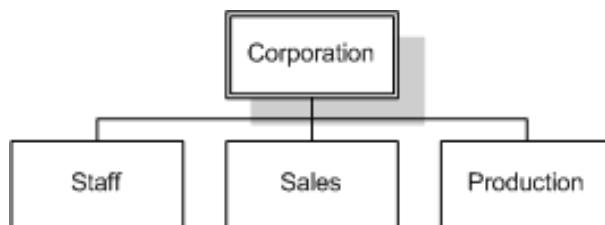
Published 2003-05-30

Clarion has a template for building a tree list control, called the RelationTree. One of the disadvantages of this control is that the maximum tree level is fixed by the number of files you use in it.

In this article, I will outline the very simple, basic idea for a non-level-limited, page loaded tree, using a standard browse control. I will also describe the Clarion basics for displaying tree levels in a browsebox, which can be applied to a hand-coded list box too.

### Standard RelationTree implementation

You can use a tree when the application has records, which can be shown in a certain logical order from a user point of view. Take for example the organization chart in Figure 1:



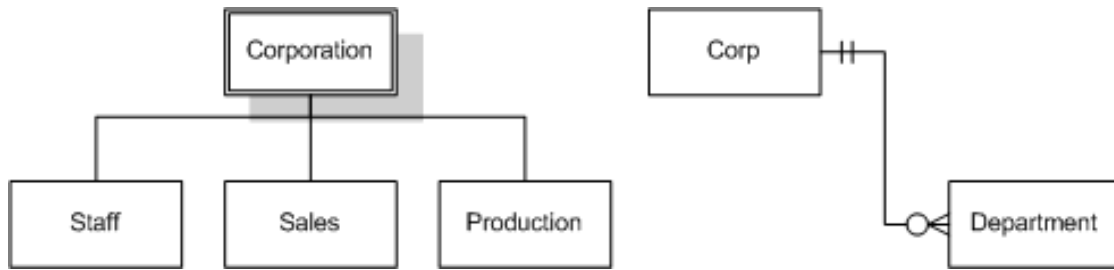
**Figure 1. A simple organization chart**

Figure 1 shows a corporation with three departments. Note that I'm not referring to employees in this organization, but to departments. The employees will come later.

To model this using the ABC tree template, you need to create a dictionary which contains at least two files: One for all corporations, and one for all



departments (see Figure 2).



**Figure 2. Org chart and the database design**

On the right side I have drawn the dictionary relationship. By the way, I use [Microsoft Visio](#) to draw my dictionaries, even before I start working on the dictionary itself.

A corporation record has zero or more departments (1:N relation), and a department record belongs to exactly one corporation record (N:1 relation). This relationship is exactly the same as an order-orderline relationship: you can't add an orderline (item) without having added an order (customer, delivery date etc) first.

But of course you can add multiple corporations, each with their own departments. And each corporation does not have to have exactly three departments.

In a window, the tree looks like Figure 3.

Corp records		Department records		
ID	Name	ID	CorpID	Name
1	Heineken	2	1	Staff
3	Grolsch	3	1	Sales
		6	3	Sales
		7	3	Expedition
		12	3	Production

**Figure 3. RelationTree control and the database record contents**

On the left is the tree, on the right are the records for both files. Note that the ID field is a primary, autonumber key field. The CorpID field is a foreign key field, and refers to the ID field in the corporation record. Note that because both corporations have a Sales department, I add this record twice, so the department name can't be a unique key.

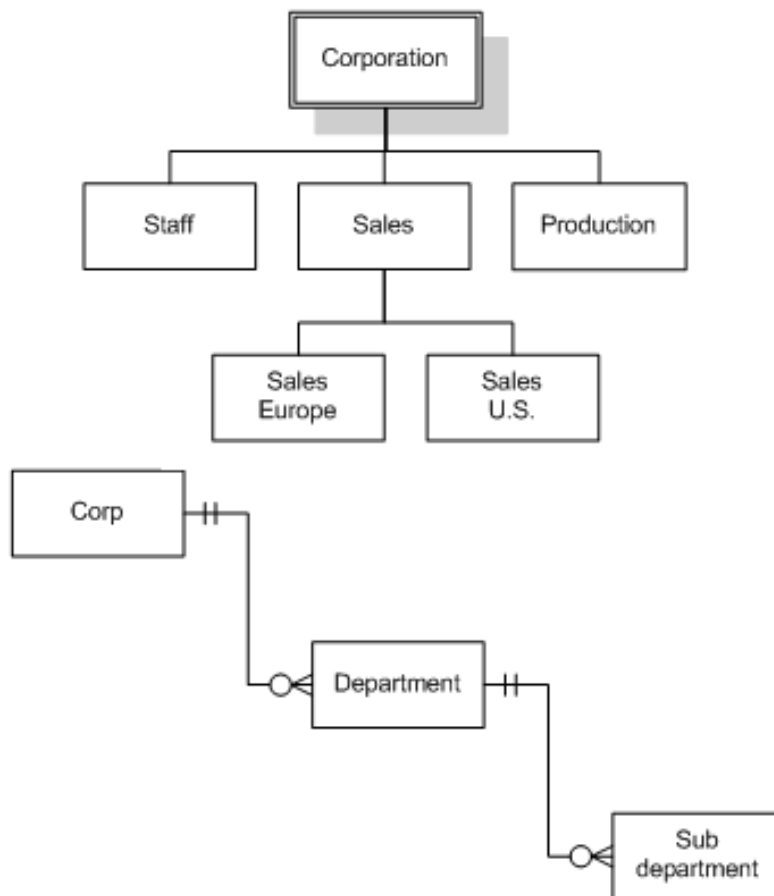
Now this all looks like fun! With the RelationTree template you define a primary

file (corp) a secondary file (department) and two update procedures, and there you go...

## Maintenance on RelationTree design

So you have your application ready and shipped. Then your client calls and says, hey, I want to enter a customer which has not only departments, but subdepartments too (1:N relation). And a subdepartment is linked to a department. In fact, a department can have zero or more subdepartments.

So, you go back to the dictionary, add a subdepartment file, and link it to the department. You also go back to the RelationTree control, and add the subdepartment file here too, as shown in Figure 4.

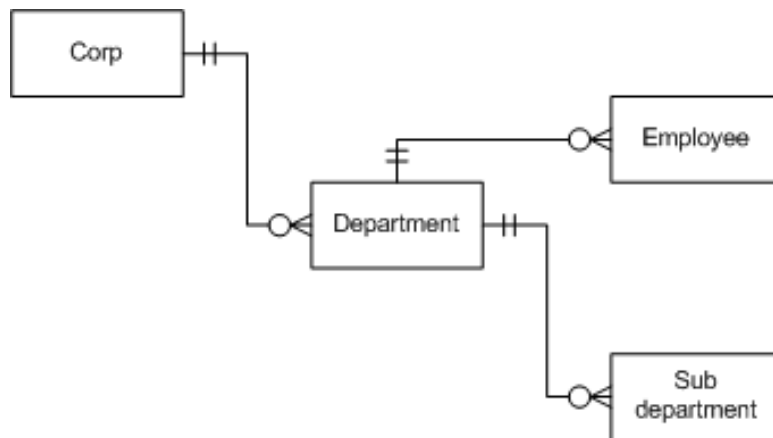


**Figure 4. Extending the org chart**

If the client wants yet one more level, you have to go back to the drawing table again. Clearly, when you provide something as flexible as an org chart, with a variable number of levels, you can't use the RelationTree control template, unless you're absolutely sure the maximum depth is fixed.

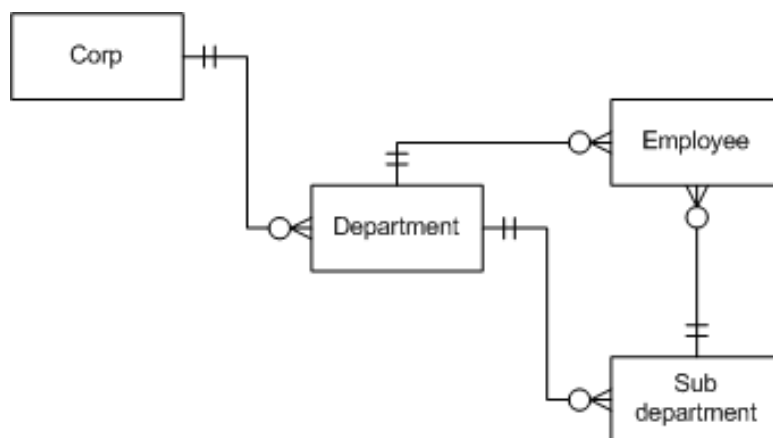
Even if the fixed levels are no problem, the RelationTree template design has another big disadvantage, which I will explain using an example.

Imagine you want to add employees. An employee works in an organization, and you want to link an employee to a department. As an employee can only work in one department, and a department can employ multiple employees, you link the employee to a department, as in Figure 5.



**Figure 5. Database design: Employees added to departments**

But, employees can also work in a subdepartment! So you need another link in the employee file to the subdepartment. Now how do you know if the employee works in a department or in a subdepartment? The employee file now needs two fields, one to reference to a department (EMP : DepartmentID), and one to reference to a subdepartment (EMP : SubDepartmentID), as in Figure 6. You may also need to write code to prevent an employee from being shown as working in a department and a subdepartment at the same time (The business rule for the update employee is that either the EMP : DepartmentID field or the EMP : SubDepartmentID cannot be zero, and that both fields can't be unequal to zero at the same time.)



**Figure 6. Database design: Employees added to departments and subdepartments**

The same story goes for invoices. You can send an invoice to a company (corp), to a department or to a subdepartment. You end up having links all over the place and lots of extra code to prevent double relations.

## Requirements

In the sections above I have shown that the standard relation tree control is not particularly flexible, and that the maximum number of levels is restricted to the number of files you have defined in the database design. So, if you want to avoid these issues, you can define the requirements as:

1. The number of levels is not tied to files defined in the design.
2. There must be only one relationship when linking a tree record to another file.
3. The file should be page loaded, preferably using a standard browse box (and browse class).

## Tree level basics

Before I can go into details about the solution, I need to show how a listbox is instructed to display tree levels. Let's go back to the basics.

A listbox control uses a queue to display its data. If you populate a browse box control, you can go to the list box formatter and add the database fields. The template generates a local queue in the procedure, with fields like your database fields. The browse class fetches records and adds them to the queue. So the queue contains content, but the queue itself knows almost nothing about how to display the content.

If your database contains a LONG field for example, which is actually a date displayed using the @D6- picture, the queue just contains a LONG field. In the listbox formatter, you set the picture, which can be @D6-, @D01 or @D02.

These and other column or field settings are translated by the IDE to a format string. A Clarion listbox gets information about how to display the queue fields by interpreting that string. The format string is assigned to a listbox by either setting `PROP:Format` or by using the `FORMAT` property on a `LIST` control in the window editor.

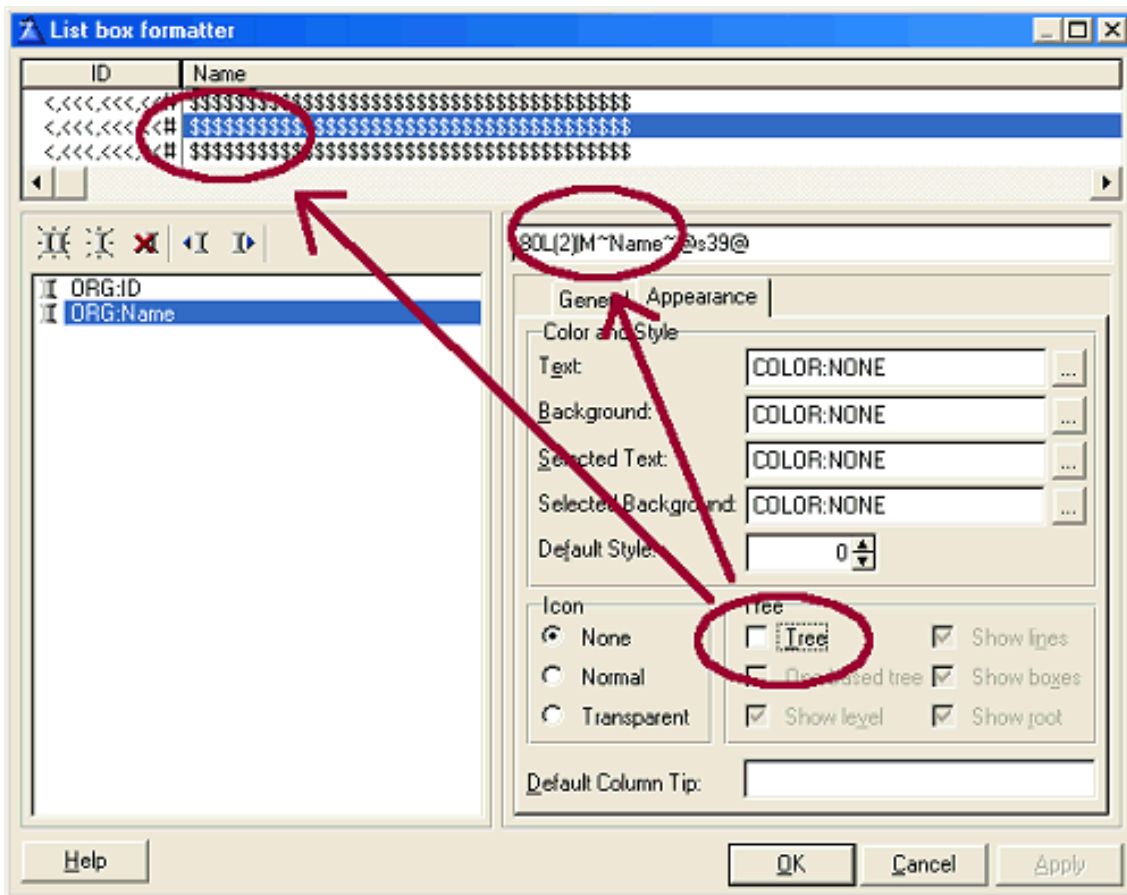
```
Window WINDOW('No title'),AT(, ,100,200)
  LIST,AT(5,5,90,190),USE(?List),|
  FORMAT(' 52L~ID~@D6-@'),FROM(Queue:Browse)
END
```

For a date column without any other settings, the minimum format string is something like this:

52L~Date~@D6~@

This string tells the list control that the width of the column is 52 (pixels or DLUs), the title is "Date" and the display picture is @D6-.

So, what about tree levels? Figure 7 shows what happens if you use a Name field, without a tree.



**Figure 7. List box formatter settings without tree levels**

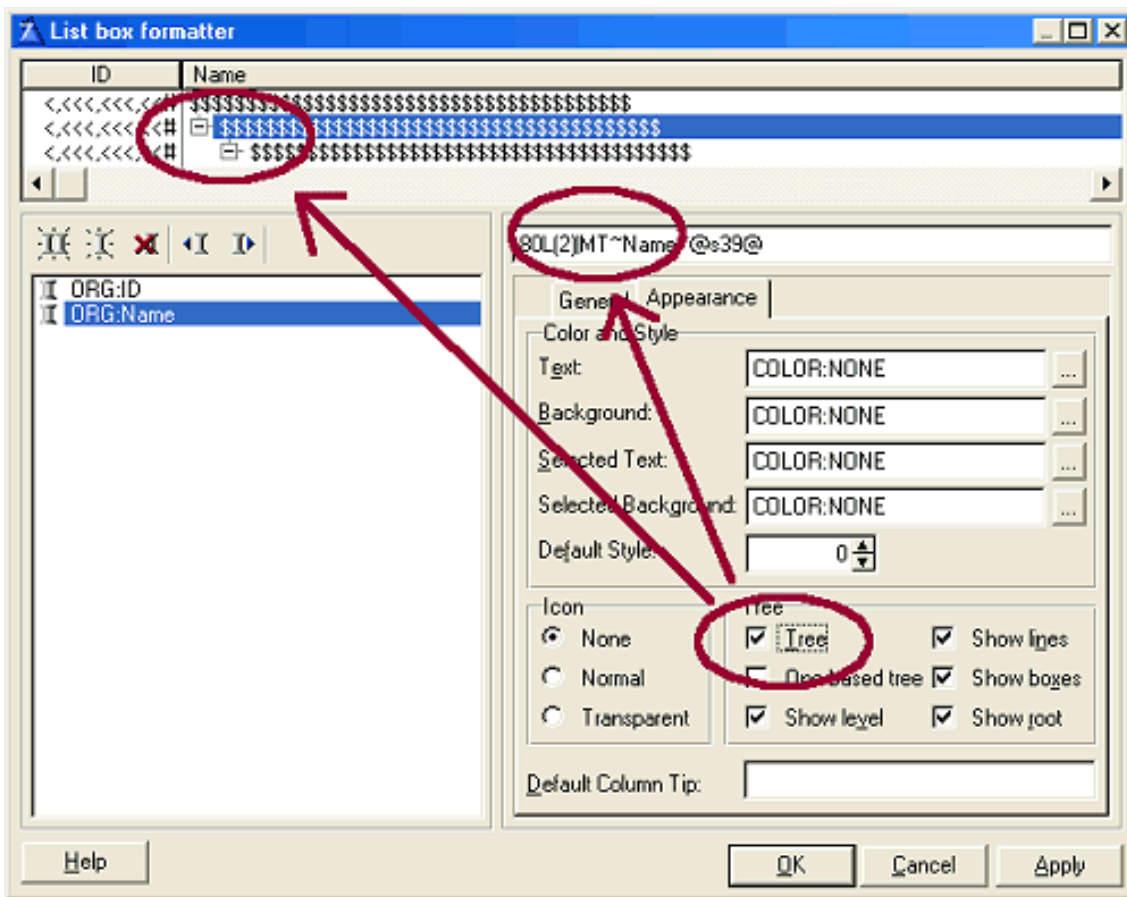
Notice that the **Tree** property is not checked, there's no tree, and the format string is 80L(2) |M~Name~@S39@. Now the generated queue is:

```
Queue:Browse:1      QUEUE
ORG:ID              LIKE(ORG:ID)
ORG:Name            LIKE(ORG:Name)
Mark                BYTE
ViewPosition        STRING(1024)
                    END
```

You see that the queue fields are named exactly after the field names in the dictionary. But remember, these are queue fields, which do not yet have any relationship with the file being browsed! It's the underlying `BrowseClass` which copies the file buffer fields to the queue fields.

Notice that while I have only added two fields in the window formatter, two more fields are added to the queue. The `Mark` field is a legacy field originally designed for tagging purposes. The `ViewPosition` is used by the `BrowseClass` as a unique reference to a record in the view.

Now, go to the listbox formatter, and check the `Tree` property. You'll see a tree emerge in the listbox, for that field. And, most important, you'll see your format string change too, as shown in Figure 8..



**Figure 8. List box formatter settings with tree levels**

Notice the "T". The listbox control in the runtime library interprets the T as: "Hey, on this field I need to display tree nodes."

Now go to the source and look at the generated queue:

```
Queue:Browse:1      QUEUE
```

```

ORG:ID                LIKE(ORG:ID)
ORG:Name              LIKE(ORG:Name)
ORG:Name_Level      LONG                !Tree level
Mark                 BYTE
ViewPosition         STRING(1024)
                    END

```

Because the listbox control needs to know the tree level of the record, it adds a long field in the queue, and adds `_Level` to the name of the field.

The listbox control now knows:

- from the format string (T) it should display nodes
- that the tree level is stored in the queue field directly following the field itself.

But where do you assign a value to the queue's `Name_level` field? Notice that it's not defined as a `LIKE` field in the database, so there's no evidence of a direct link being made by the templates for the `BrowseClass`. There's no template prompt in the `BrowseClass` for this level field. So you have to go back to an embed in the generated code.

Remember that the `BrowseClass` fetches a record and copies the used fields from the file buffer to the queue? This is actually done by the `BrowseClass.SetQueueRecord` method. In the `BrowseClass` embed, **SetQueueRecord, after parent call**, you need to set the tree level queue field. Here's the source:

```

BRW1.SetQueueRecord PROCEDURE
! Start of "Browser Method Data Section"
! [Priority 5000]
! End of "Browser Method Data Section"
CODE
! Start of "Browser Method Code Section"
! [Priority 1300]
! Parent Call
PARENT.SetQueueRecord
! [Priority 5500]
Queue:Browse:1.ORG:Name_Level = 1 ! Set all levels to 1
! End of "Browser Method Code Section"

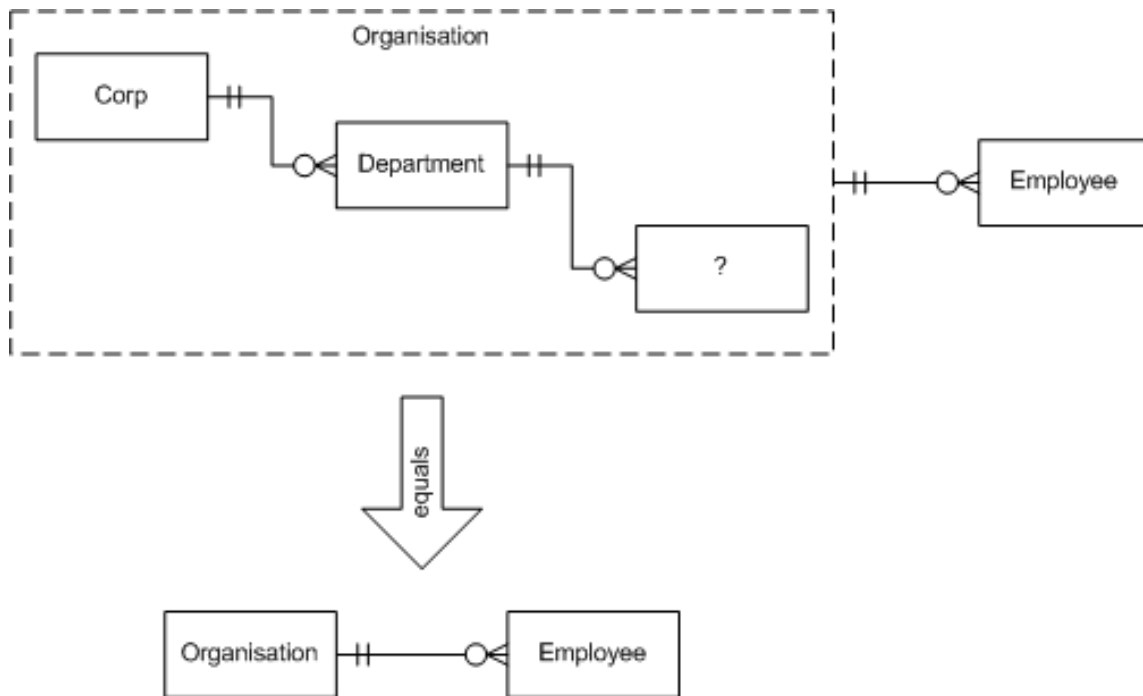
```

Now of course you won't set all levels to 1, but to the level needed. So it's time to look at the final page loaded, browse tree solution.

## A maintainable tree solution

In the org chart described earlier, you have seen that modeling different levels in

separate files is not an option. You want the content (all levels of all corporations) in a single file. You also want this because then you can easily link a single file to other files (see Figure 9).



**Figure 9. Merging all organization levels in one file**

Here, the organization file contains corporation, department and subdepartment records (and more levels if needed).

Now you have two problems:

1. An organization record must know its level
2. When displaying the records, a the right order must be preserved: A department record should be displayed below it's corporation record.

To solve this, you add a single STRING (or CSTRING) field to the organization record. Nowadays I prefer CSTRING fields, because you can concatenate them without using the CLIP ( ) function.

As I always add a unique ID, I have a minimum record layout like this:

```

Organization FILE,PRE(ORG)
ID            ULONG
Name          STRING(20)
SeqNo        CSTRING(255)
END

```



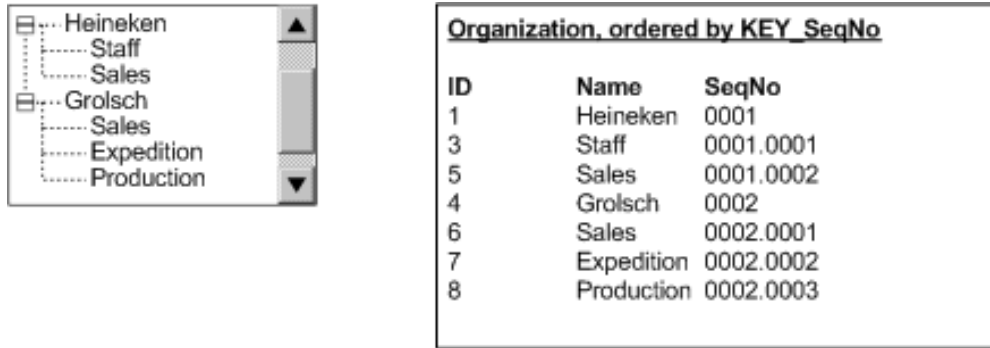
And I define three keys:

KEY\_ID (Primary key with autonumber)

KEY\_Name (Non-Unique key, excluding empty keys)

KEY\_SeqNo (Non-Unique key, excluding empty keys)

Now, let's go back to the Grolsch-Heineken example, as shown in Figure 10.



ID	Name	SeqNo
1	Heineken	0001
3	Staff	0001.0001
5	Sales	0001.0002
4	Grolsch	0002
6	Sales	0002.0001
7	Expedition	0002.0002
8	Production	0002.0003

**Figure 10. Organization content**

Notice that the ID fields don't match the example given earlier, but as the ID field is just a unique field which has nothing to do with the tree structure itself, you can ignore that completely. Also notice the special contents of the SeqNo field. These contents provide all that's necessary to maintain the order in the file, and to calculate each level.

Typically, an end-user won't see the SeqNo field content.

Notice also that corporate levels use only four characters: '0001' and '0002'. The department levels add five characters to that (Grolsch.Sales = '0002.0001'). How do you calculate each level? The length of the string gives the level in the tree:

Queue:Browse:1.ORG:Name\_Level = INT(LEN(ORG:SeqNo) / 5) + 1

I use the INT( ) function because the queue field is a long, which can only contain integer numbers. This way, the "corporate" level is

$\text{INT}(\text{LEN}(\text{'0001'}) / 5) + 1 = 1$  and a "department" level is

$\text{INT}(\text{LEN}(\text{'0001.0001'}) / 5) + 1 = 2$ .

An advantage of this solution is that when I want to process all departments of 'Grolsch', I set a filter to process all records where the first four characters are '0002', because they all are departments or sub(sub(sub))departments of 'Grolsch'.

Unfortunately, there are some disadvantages too:

If you choose to use four characters (and a fifth for the '.', which is only added for readability by the way) you can add a maximum of 9999 organization records. If you use five characters, you can add 99999 records (you could go hexadecimal to get more levels). In return for this maximum restriction, the tree level is highly flexible, depending on the length of the SeqNo field:

With 255 characters (256-1 for '<0>', because it's a CSTRING), you can go  $255/5 = 51$  levels deep, but you still link other files, like employees, to a single file in the dictionary.

After a long story and a lot of theory, it's time for implementation.

First define a file, with a SeqNo field (or whatever you call it), and create an ascending key for this field.

## The Browse

Create a procedure with a browse box and populate the name field. Be sure to check the **Tree** property on the **Appearance** tab in the listbox formatter.

Now go to the embeds, and check the name of the Name\_Level field in the generated queue. Normally, this is Queue:Browse.Name\_Level.

Scroll down or find the SetQueueRecord method for the browseclass. After the parent call, add the line:

```
Queue:Browse.Name_Level = INT(LEN(ORG:SeqNo) / 5) + 1
```

When you add a record, it should appear below the currently selected record, so you need to pass the current SeqNo field (pass by value) to the update procedure.

Here comes a little problem: The BrowseUpdateControls template generates different code for a standard update call, or a call to a procedure when passing a parameter.

If you don't use a parameter, the BrowseClass.Ask method is called internally, and here, a new record is primed (autonumbered key fields are set).

If you use a parameter to call the update, the BrowseClass.Ask method is not called, and autonumbered key fields are not set. In fact, the fields of the new

record are not even cleared.

And you do want to pass a parameter, because new records should know what their parent is in the tree. In fact, you want to prime the `ORD:SeqNo`, based on the contents of `ORD:SeqNo` from the parent.

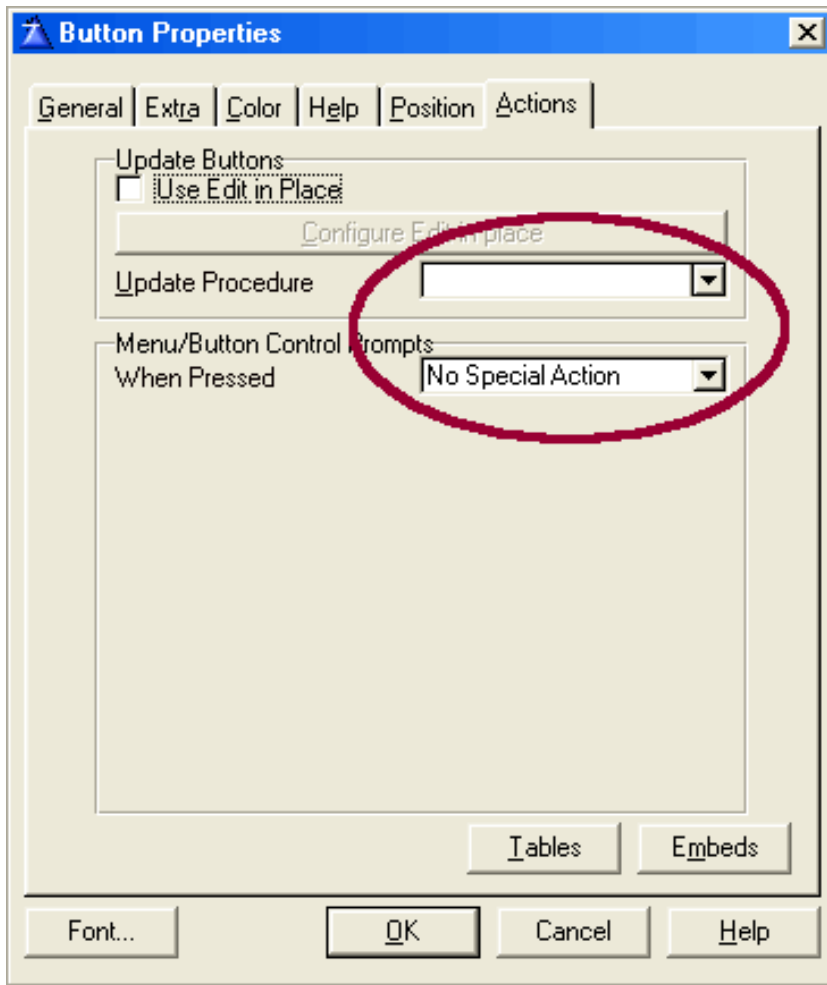
You need to write some code, at the **?Insert, accepted** embed, after generated code:

```
OF ?Insert
  ThisWindow.Update
  ! Start of "Control Event Handling"
  ! [Priority 6000]
  SEQNo = ORG:SeqNo
  IF Access:Organization.PrimeRecord() <> LEVEL:Benign
    CYCLE
  END
  Req          = InsertRecord
  GlobalRequest = Req
  UpdateOrganization(SeqNo)
  Response     = Globalresponse
  BRW1.ResetFromAsk(Req, Response)
  ! End of "Control Event Handling"
```

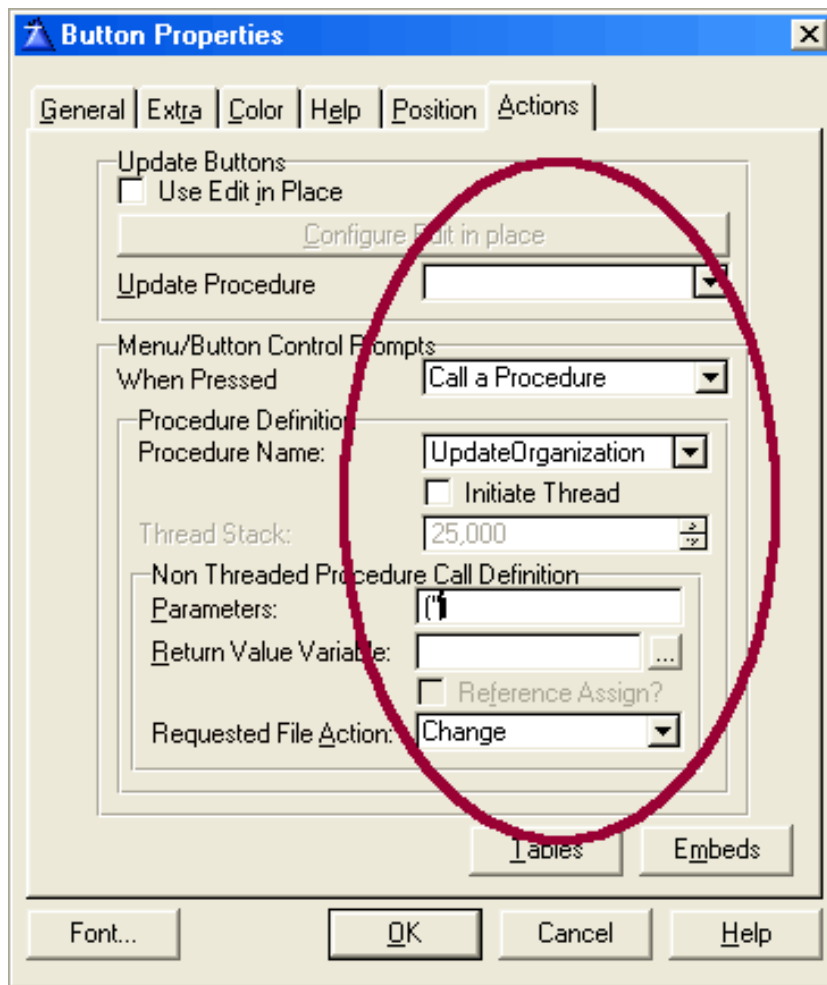
This code is taken more or less from the `BrowseClass.Ask` method (`\libsrc\abbrowse.clw`), to prime autonumbered keys. The `ResetFromAsk` method is **PROTECTED**, so normally you can't use it. Unfortunately you must go to `libsrc\abbrowse.inc` and remove the **PROTECTED** attribute. Also, because the `BrowseClass.ResetFromAsk` method takes variables passed by address, you need to define these two variables locally. You also want to store the current (parent) `ORD:SeqNo` field temporarily:

```
ThisWindow.TakeAccepted PROCEDURE
ReturnValue          BYTE,AUTO
! Start of "WindowManager Method Data Section"
! [Priority 3500]
Req                 BYTE,AUTO
Response           BYTE,AUTO
SeqNo              LIKE(ORG:SeqNo)
```

Now that you have written the embedded code, you configure the `BrowseUpdateButtons` template to call the update procedure. Go to the browse window in the window formatter, add the update buttons, and go to the Actions tab of the update buttons, as shown in Figures 11 and 12. Do *not* use the Update Procedure prompt; instead, select **Call a Procedure**.



**Figure 11. Configuring BrowseUpdateButtons for Insert**



**Figure 12. Configuring BrowseUpdateButtons for Change and Delete**

Select the requested file action (Insert/Change/Delete) and fill out the prompts as shown. Unfortunately, you have to do this for all three update buttons separately.

Notice that you only need to pass ORD:SeqNo contents if you insert a record. But because the update procedure expects a string to be passed, you set the parameters for the change and delete action to "" (empty string).

Now, the browse is ready.

## The update procedure

As the update procedure receives a string, you need to change the prototype. Open the update procedure and type at the prototype prompt:

```
(STRING SeqNo)
```

Copy this to the parameter prompt too, so they both contain the same text.

In the update procedure go to the **WindowManager**, **Primefields** embed and

add this code:

```

ThisWindow.PrimeFields PROCEDURE
! Start of "WindowManager Method Data Section"
! [Priority 5000]
NewSeqNo ULONG(1)
CurSeqNo LIKE(ORG:SeqNo),AUTO
SaveID USHORT,AUTO
! End of "WindowManager Method Data Section"
CODE
! Start of "WindowManager Method Executable Code Section"
! [Priority 3800]
CurSeqNo = ORG:SeqNo
SaveID = ACCESS:Organization.SaveFile()

ORG:SeqNo = ORG:SeqNo &' .9999 '
SET(ORG:KEY_SeqNo, ORG:KEY_SeqNo)
IF ACCESS:Organization.Previous() = LEVEL:Benign AND |
    SUB(ORG:SeqNo,1,LEN(CurSeqNo)) = CurSeqNo THEN
    IF LEN(ORG:SeqNo) <> LEN(SeqNo) THEN
        NewSeqNo = SUB(ORG:SeqNo, LEN(ORG:SeqNo)-3,4) + 1
    END
END

ACCESS:Organization.RestoreFile(SaveID)
ORG:SeqNo = CurSeqNo & CHOOSE(CurSeqNo='',','.') |
    & FORMAT(NewSeqNo,@N04)

```

Basically this code defines your own autonumber method. The `SaveFile()` and `RestoreFile()` calls are needed because a new record has already been added to the file because of the autonumber on `ORG:ID`.

## Conclusion

The RelationTree template is not designed to contain a flexible number of levels in a tree, and database design is difficult when a single file links to multiple levels.

The solution described above is more flexible, but requires a lot of embedded code. The solution is not perfect yet, because if you delete a level, sublevels must be modified or be deleted too. In my next article, I will show you how to delete records, and write a template, based on the code I've described here.

[Download the source](#)

---

*[Ronald van Raaphorst](#) studied Chemical Engineering at the University of Enschede (UT), the Netherlands. But he found programming was more fun than designing a chemical plant, and when a roommate asked him to help*

*start a software company, he found the choice easy to make. Ronald has used Clarion since 1994, beginning with Clarion 3 for DOS. Compad Software, which he co-owns, sells software to a small group of bakeries, he spends a considerable amount of time on the phone helping users, finding (and creating) new bugs, writing manuals, and of course programs. Ronald is in charge of developing Compad's products, and his colleague is on the road selling.*

## Reader Comments

[Add a comment](#)

**Great article. Dumb question -- In the article you...**

**I guess that is a mistake then. Remove either Protected...**

**My mistake - I actually started to correct that in the...**

Copyright © 1999-2003 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.