

# Clarion Magazine



Please note that Clarion Magazine is temporarily publishing three issues per month rather than four so we can devote more time to our upcoming [book series](#). All subscriptions are automatically extended.



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

Articles: 

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

News: 

*Posted Tuesday, December 02, 2003*

## [News](#)

### [NAME\(\) Becomes Irrelevant](#)

NAME has been an important attribute of the FILE statement for a long time. In Clarion 6, however, there is a new feature that makes PROP:NAME irrelevant in many situations, as Henry Plotkin shows.

*Posted Thursday, December 04, 2003*

[thetingroup.com Clarion 6 Compatibility](#)

[1st Logo Design Partnership Program](#)

### [Moving Applications to Oracle: RI And AutoNumbering Part 3](#)

In this third installment Jon Waterhouse looks at transactions and concurrency control in Oracle.

*Posted Friday, December 05, 2003*

[gNotes 3.0 Released](#)

[Jesus Moreno and Gitano Software Named Third-party Vendor of the Week](#)

### [A Simple Versions And Annotations Template](#)

Recently Ronald van Raaphorst realized he needed some kind of annotation system for his Clarion app. The result was this template that lets a developer record changes in any procedure, and print an overview of all annotations per procedure or application.

*Posted Friday, December 05, 2003*

[Clarion Developers](#)

[Challenge Week 16 Winners](#)

[Simssoft C6 Compatibility](#)

[CapeSoft Tip of the Week](#)

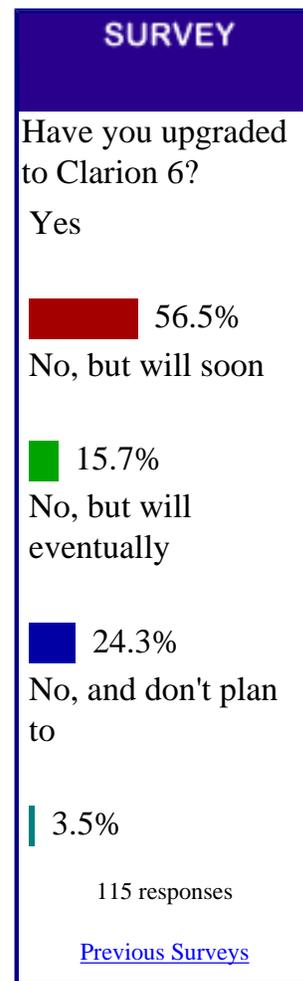
[BST 2.5](#)

[Microsoft Demos and Downloads](#)

### [Weekly PDF For December 1-6, 2003](#)

All articles for December 1-6, 2003 in PDF format.

[DeveloperPLUS/Lodestar](#)



## **One Year Ago In CM**

[PDF for December, 2002](#)

[The Clarion Advisor: Displaying Clarion Dates In Excel](#)

[DNA for Clarion: Manipulating Browse](#)

Posted Sunday, December 07, 2003

## [Clarion 6 First Look: The IDE](#)

In this concluding part to Clarion Magazine's first look at C6, David Harms goes through the Gold release CD install, and reviews the C6 IDE improvements.

Posted Thursday, December 11, 2003

## [Moving Applications to Oracle: RI And AutoNumbering Part 4](#)

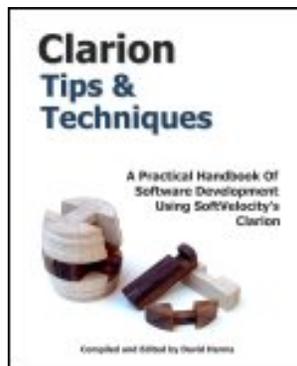
In this conclusion to his series on Oracle transactions and relational integrity, Jon Waterhouse discusses transaction isolation levels.

Posted Thursday, December 11, 2003

## [Pre-Order Clarion Tips & Techniques Book Now](#)

You can now pre-order Clarion Magazine's Tips & Techniques book for \$20 off the regular price.

Posted Wednesday, December 17, 2003



## [Weekly PDF For December 7-13, 2003](#)

All articles for December 7-13, 2003 in PDF format.

Posted Wednesday, December 17, 2003

## [Clarion Magazine Holiday Hours](#)

The Clarion Magazine office will be closed from December 23 through January 4. We will be checking email periodically over the holidays, but it may be a couple of days before you get a reply. Book and subscription orders will continue to be processed automatically. There will also be one more issue published right at the end of the month.

Posted Tuesday, December 23, 2003

## [Save and Restore Window Size and Location](#)

## [Software Holiday Schedule](#)

## [Clarion Handy Tools](#)

### [O8A1.4](#)

## [Microsoft ISV Program](#)

## [Lost Editor Window Finder Utility](#)

## [Fenix Newsletter #1](#)

## [xXPpopup 1.1](#)

## [MAV Direct ODBC Versions 005 And 002](#)

## [Informatic Consulting Named Third-party Vendor of the Week](#)

## [Clarion Developers Challenge Week 15 Winners](#)

## [Icetips December 2003 Newsletter](#)

## [ClarionNET Available From ClarionShop](#)

## [Clarion Developers Challenge Week 14 Winner](#)

## [#cw-talk Back Up](#)

## [xFunction Library 2.0](#)

## [See Pictures Of Our New Clarion Book](#)

## [Tips & Techniques Book On Sale December 15](#)

## [Cells With A VLBPROC \(Part 2\)](#)

## **Two Years Ago In CM**

Jay Guengrich - automation

## [Clarion Magazine To Resume Publication In January](#)

## [The Clarion Advisor: Sizing Windows](#)

## **Three Years Ago In CM**

## [Clarion News - January 2001](#)

## [Completely Dynamic Report Orders and Breaks Part 2](#)

## [Interfacing Satellite Forms Applications and Clarion for Windows](#)

## **Four Years Ago In CM**

## [How To Handle Additional Sort Orders](#)

## [The SQL Answer Cowboy](#)

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

Nardus Swanevelder goes on a template-writing quest [EasyMultiTag 2.06](#) to blend window size and location saving with the IceTips Cowboy SQL Templates and Ingasoftplus's Easy Resize and Split templates.

*Posted Wednesday, December 31, 2003*

### [Creating Utilities For MS SQL 2000](#)

Clarion's PROP:SQL is an often under-appreciated language statement. As Bernie Groperrin shows, you can use PROP:SQL for lots of things, including creating SQL utilities for MSSQL. This task is made even easier with Dan Pressnell's clever "Query from Queue" mechanism.

*Posted Wednesday, December 31, 2003*

### [Russ's UK Trip](#)

A few photos from the recent Clarion UK gathering with Russ Eggen in London.

*Posted Wednesday, December 31, 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.

[SetupBuilder 4.03 C6 Compatible](#)

[Firebird ODBC Driver](#)

[kbalertz.com Indexes MS KB](#)

[ClassViewer 6.0 Version 2003.12.03](#)

[AFE for C6Gold\(ABC\)](#)

[Capesoft Software Named Third-party Vendor Of The Week](#)

[BIT, BOT, BST Updated For C6](#)

[ADDA Release Candidate](#)

[BigTamer Christmas Special](#)

[1st Logo Design Holiday Sale](#)

[DET For C6](#)

[Clarion Developers Challenge Week 13 Winner](#)

[BoxSoft C6 Templates Released](#)

[Clarion API Database Updated](#)

[IMPEX 5.3](#)

[Lee White and Lodestar  
Software Named Third-party  
Vendor of the Week](#)

[Icetips Reporter For C6](#)

[xPictureBrowse 2.3](#)

[SetupBuilder 5 Special Offer  
To Expire](#)

[UK Clarion Meeting](#)

[BoxSoft C6 Compatibility](#)

[MAV Direct ODBC  
Template Version 0.01](#)

[Clarion Developers  
Challenge Week 12 Winner](#)

[PowerXP Theme](#)

[New Simshape and Simsoft  
Versions](#)

[RADFusion Shopping Cart  
Goes Live](#)

[ClarionForge Unavailable  
Until Further Notice](#)

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

# Clarion Magazine

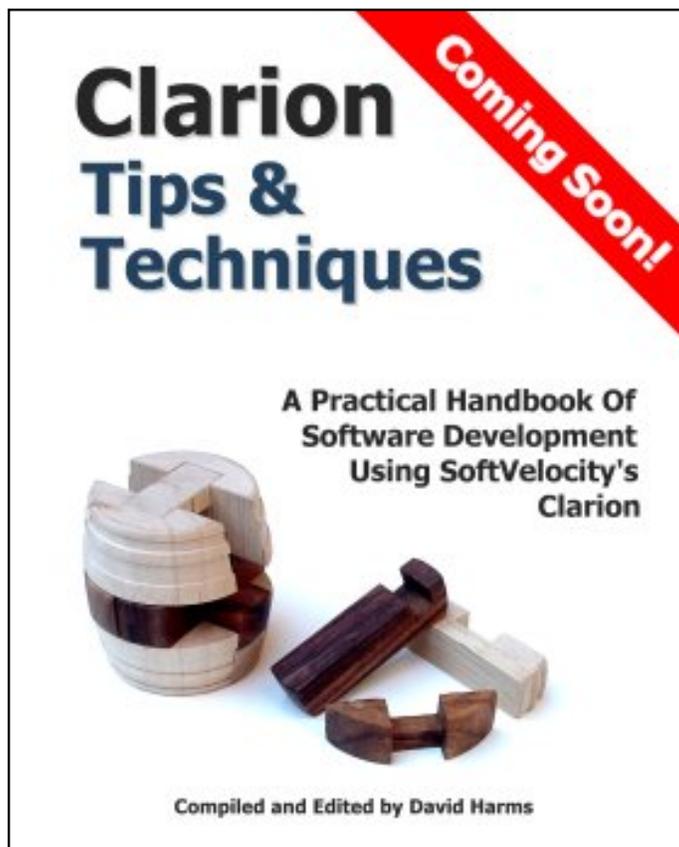
online sales and delivery  
for your applications & tools

Developer **PLUS**

## Clarion Books

**Pre-Order the Clarion Tips & Techniques book and SAVE \$20!**

We're still doing final edits and indexing the *Clarion Tips and Techniques* book, and expect it to begin shipping in January. This is a volume of approximately 600 pages, containing selected articles from Clarion Magazine's first five years in publication, as well as a small number of articles originally published in Clarion Online. These articles were written for Clarion 5.0 and 5.5, but many are just as applicable to Clarion 6. The list price is \$89.95, but if you **pre-order now**, you can get it for **just \$69.95**, a savings of \$20. This offer will end once the book is ready for printing, and although that will be soon, we don't know the exact date yet. So get your order in now!



**Pre-Order** your copy of *Clarion Tips*

and

*Techniques* now for **just \$69.95!**

[View the provisional table of contents \(PDF\)](#)

**Subscription Not Required:** The current purchase process requires you to **log in** if you have an **existing ClarionMag user ID**, or **register a new (free) ID**. Apparently this has given some people the impression that you need to be a subscriber to buy the book. This is most emphatically *not* the case! I need your address in order to calculate freight and ship you the book, and the easiest way to do this is to simply have you use the existing means of registering with Clarion Magazine, which has the added benefit of giving you access to certain [free articles](#), as well as making it easy for you to sign up for our double opt-in mailing lists. I regret any confusion this may have caused - if it appears this will be a problem I'll

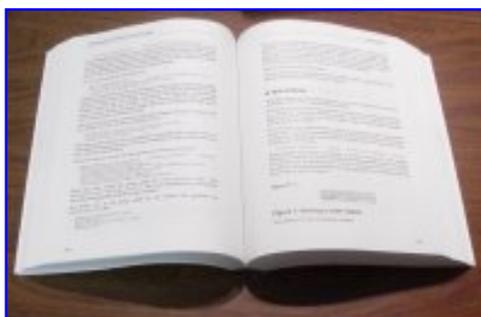
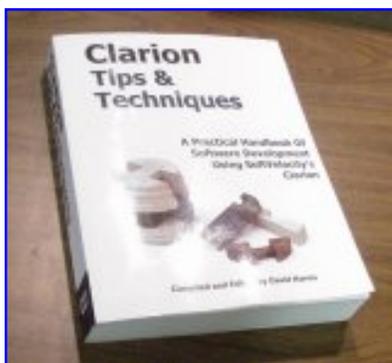
certainly look at amending the order process so you don't have to register a user id.

**Multiple Copies:** Several people have asked me how to purchase more than one copy of the book at one time. Until such time as I've made the necessary changes to the shopping cart, you can use these links: [order 2 copies](#) - [order 3 copies](#) - [order 4 copies](#) - [order 5 copies](#). The shopping cart will still indicate that the book has been added and there is one item in the cart, but when you go to the checkout or to edit the cart you'll see that the quantity is as specified.

**Refund policy:** You can, of course, get a full refund on your book order at any time up until the book ships, and if you are not satisfied with the book you have 30 days from shipment date to return the book, in new condition, to CoveComm Inc, publisher of Clarion Magazine, for a full refund or credit of the book purchase price.

## The Proof Copy

The first proof copy is back from the printer! Click on the images below for larger pictures. We still have some changes to make - the cover needs some tweaking, and of course we have to complete the index, so the book isn't shipping yet. We expect to finish production and begin shipping in January.



**To be reminded when the book ships**, enter your email address below. And check this page regularly for updates, including an upcoming table of contents and a PDF excerpt!

We are using a major US printing house to produce these perfect bound, soft cover books. As you can see from the pictures above, this is a book just like you'd expect to find on the shelves at your local computer bookstore.

**Notify me when this book is shipping!**

Email address:

You may also want to use our [RSS news feed](#) to track the book's progress. We'll be posting news items as the book's production schedule progresses.

**Why is Clarion Magazine printing books?**

Clarion Magazine, the online publication, is very popular with Clarion developers, and we're proud of the massive library of information we've published in HTML and PDF form. So why are we now printing books? Because as handy as HTML and PDF documents are, for many of us they still aren't as convenient and, well, enjoyable to read as a printed, bound book.

We are continuing to convert Clarion Magazine's HTML pages to book form. There is a stunning amount of material to choose from; in a 7.5" x 9.25" book format, the Clarion Magazine web site contains some 6500 pages of Clarion articles. That's about two feet of shelf space! If there is sufficient interest we will consider publishing all of this material, but it would be impossible to bring it all to press in a short period of time. As a result, we're focusing our efforts on areas likely to be of most immediate interest to Clarion developers. Stay tuned for more book news! And if you have any questions just send us an [email](#).

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



## NAME() Becomes Irrelevant

by Henry Plotkin

Published 2003-12-04

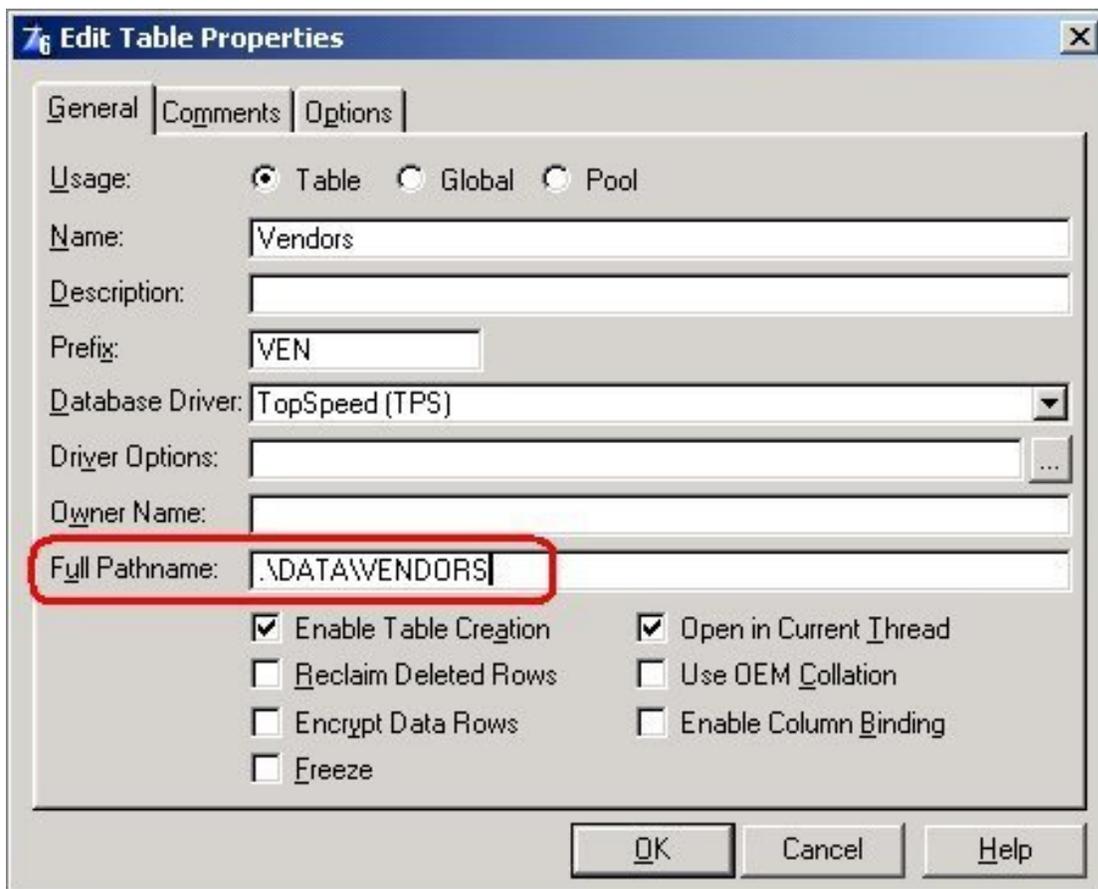
NAME has been an important attribute of the FILE statement for a long time. The NAME attribute lets me "Set [the] DOS filename specification (PROP:NAME)." In Clarion 6, however, there is a new feature that makes PROP:NAME irrelevant in many situations. I'll explain that shortly; first, a recap of how file naming works.

If empty, a file's NAME defaults to its LABEL. So, using the declaration:

```
Vendors      FILE, DRIVER( ' TOPSPEED' ), PRE( VEN ), CREATE, BINDABLE, THREAD
VendorKey    KEY( VEN:Name, VEN:ContactPerson ), DUP, NOCASE, OPT
Record       RECORD, PRE( )
Name         STRING( 40 )
ContactPerson STRING( 40 )
            END
            END
```

there will be a file, VENDORS.TPS, in the current directory (the EXE directory or the directory specified in the Windows' Shortcut "Start In") at run time.

To force VENDORS.TPS into a directory called "DATA" below the current directory, I complete the Full Pathname prompt in the dictionary (see Figure 1).

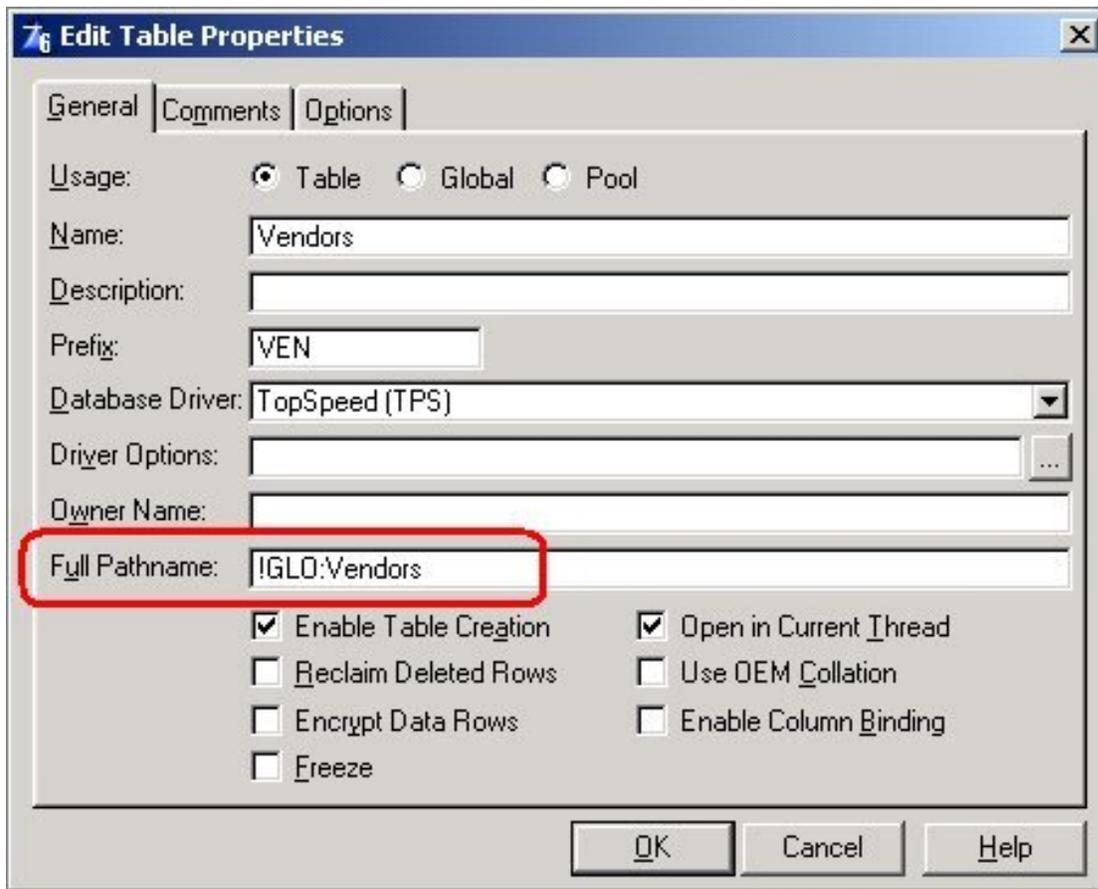


**Figure 1. Using Dictionary Editor to specify the data path**

This results in the following file declaration:

```
Vendors    FILE, DRIVER( 'TOPSPEED' ), NAME( '.\DATA\VENDORS' ), |
           PRE( VEN ), CREATE, BINDABLE, THREAD
```

For more flexibility than hard coding like this, I use a variable (see Figure 2),

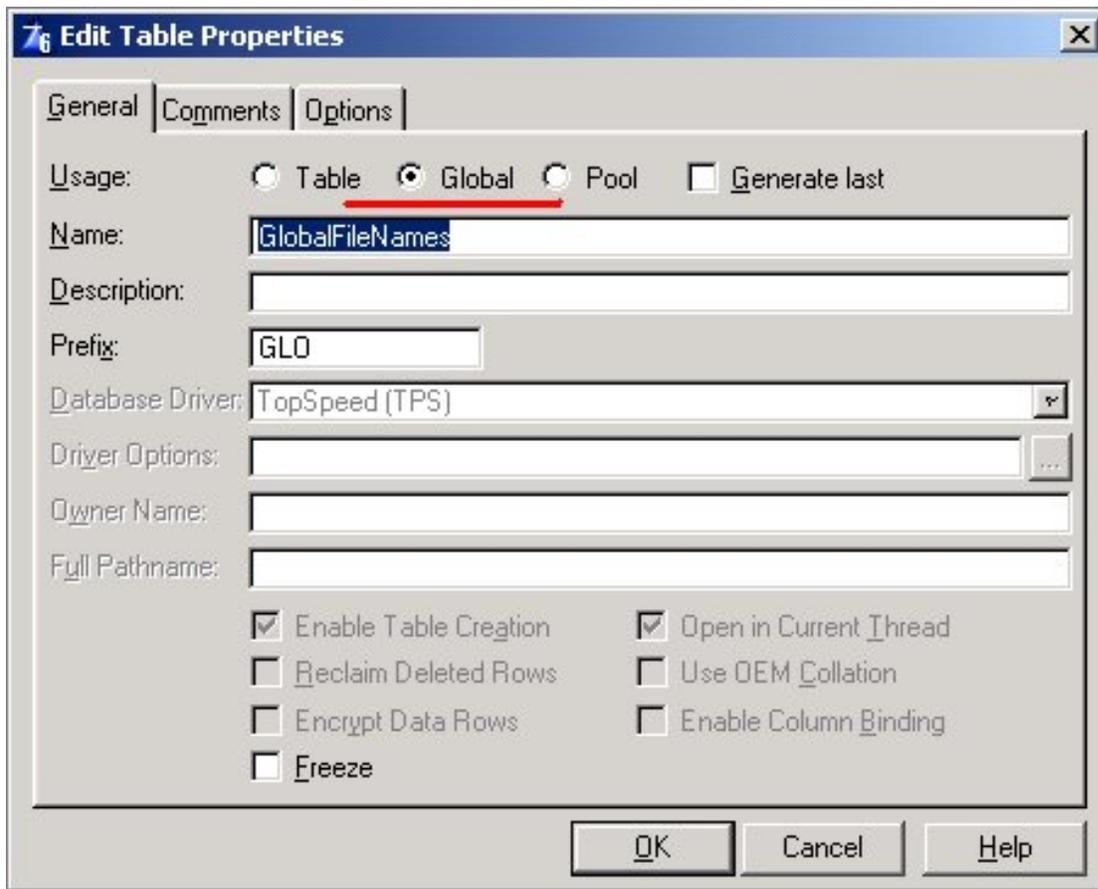


**Figure 2. Using a variable for NAME**

Using the variable results in this file declaration:

```
Vendors      FILE , DRIVER ( ' TOPSPEED ' ) , NAME ( GLO : Vendors ) , PRE ( VEN ) , |
CREATE , BINDABLE , THREAD
```

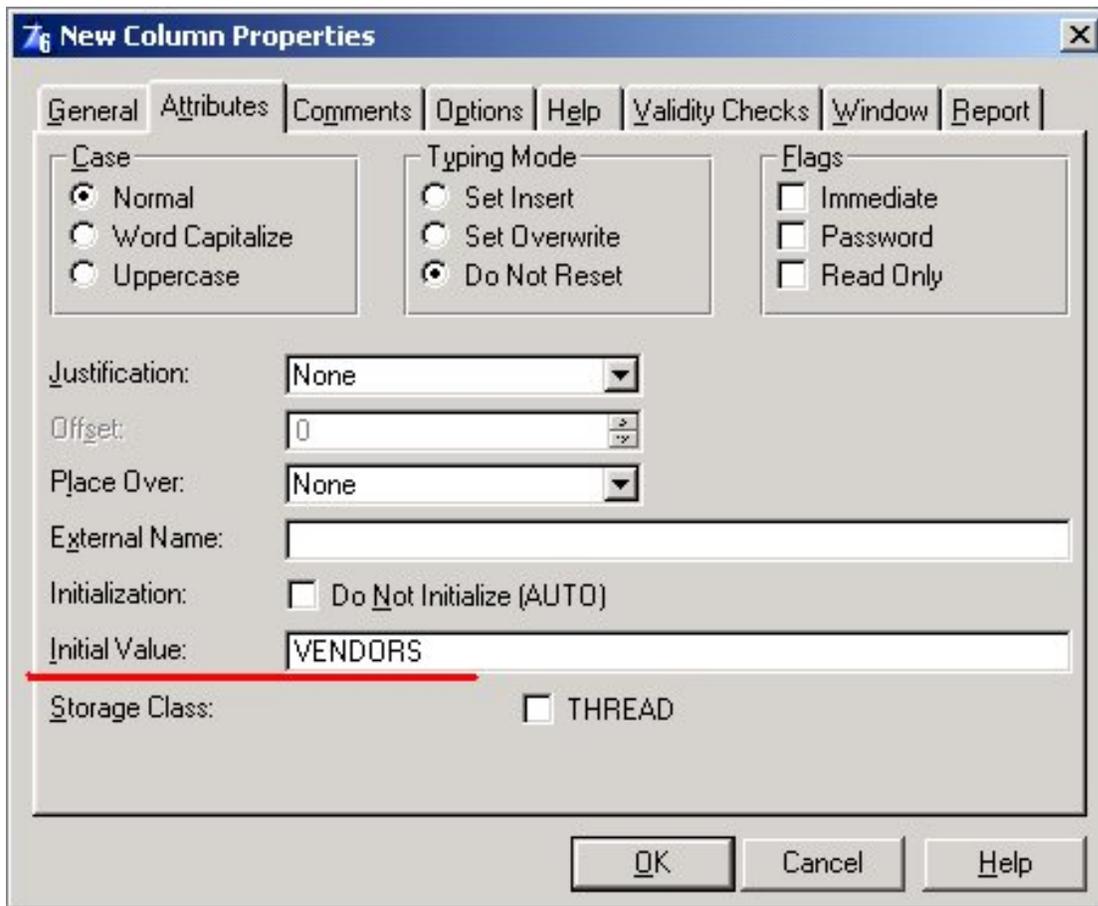
The Dictionary Editor supports using variables by allowing declaration of a global "file" containing the variables used as NAMES:



**Figure 3. Global NAME declarations in dictionary**

By declaring these variables in the dictionary, they are correctly declared in both EXEs and DLLs.

I was taught to use the dictionary to prime the value of a variable like `GLO:Vendor s` (see Figure 4).



**Figure 4. Priming a NAME variable**

Thus, even with no program action, `GLO:Vendors` contains "VENDORS." Since the driver supplied the extension, the application will find or create the file.

I was also taught to declare a global variable to contain the path (`GLO:DataPath` in Figure 5).

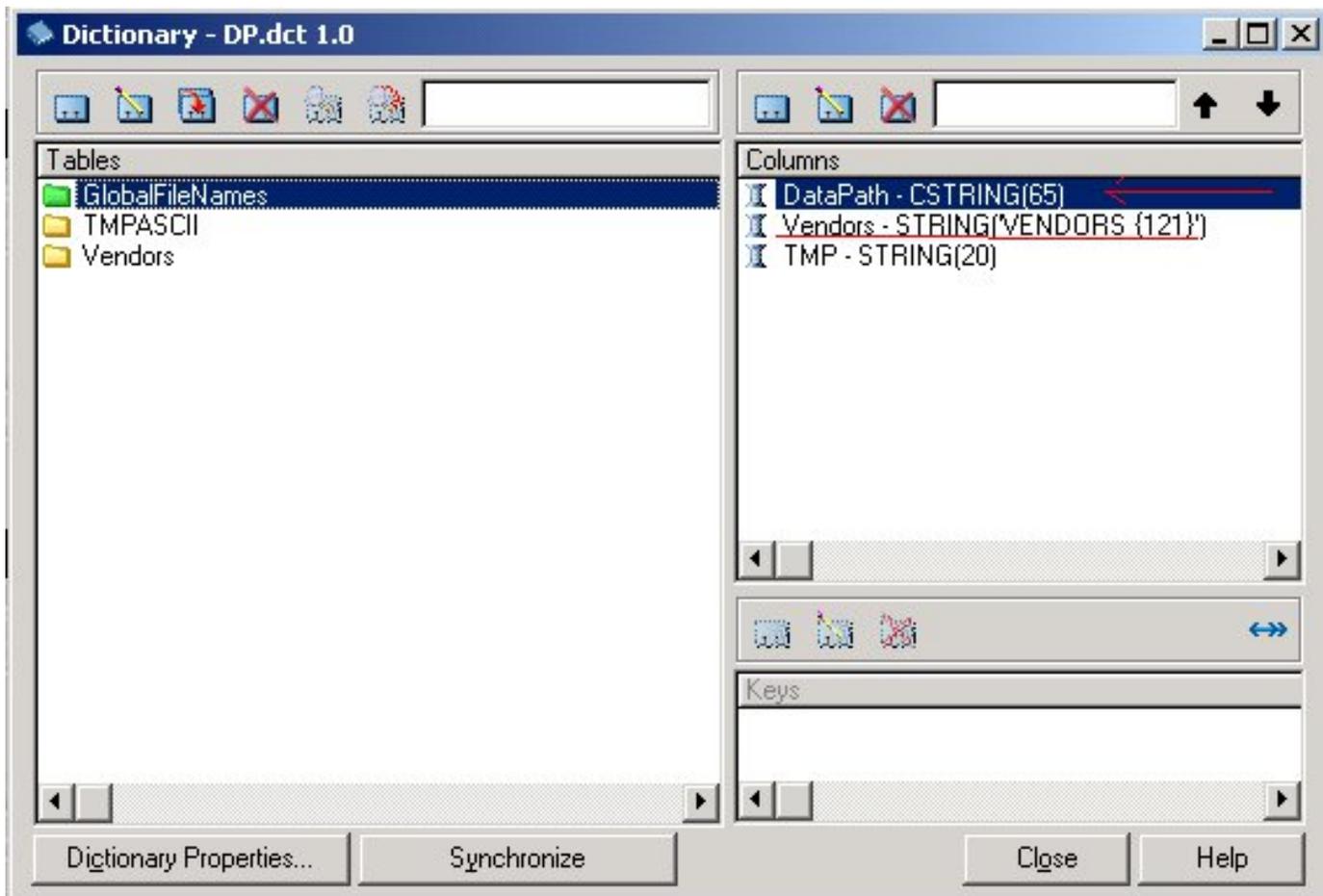


Figure 5. Global path variable

At program startup, the application queries a file and assigns a value to GLO:DataPath:

```
GLO:TMP = 'datapath.fil'
Access:TMPASCII.Open()
Access:TMPASCII.UseFile()
Get(TMPASCII,1)
GLO:DataPath = TMP:Data
```

Then, for each file, code similar to:

```
GLO:Vendors = GLO:DataPath & Clip(GLO:Vendors)
```

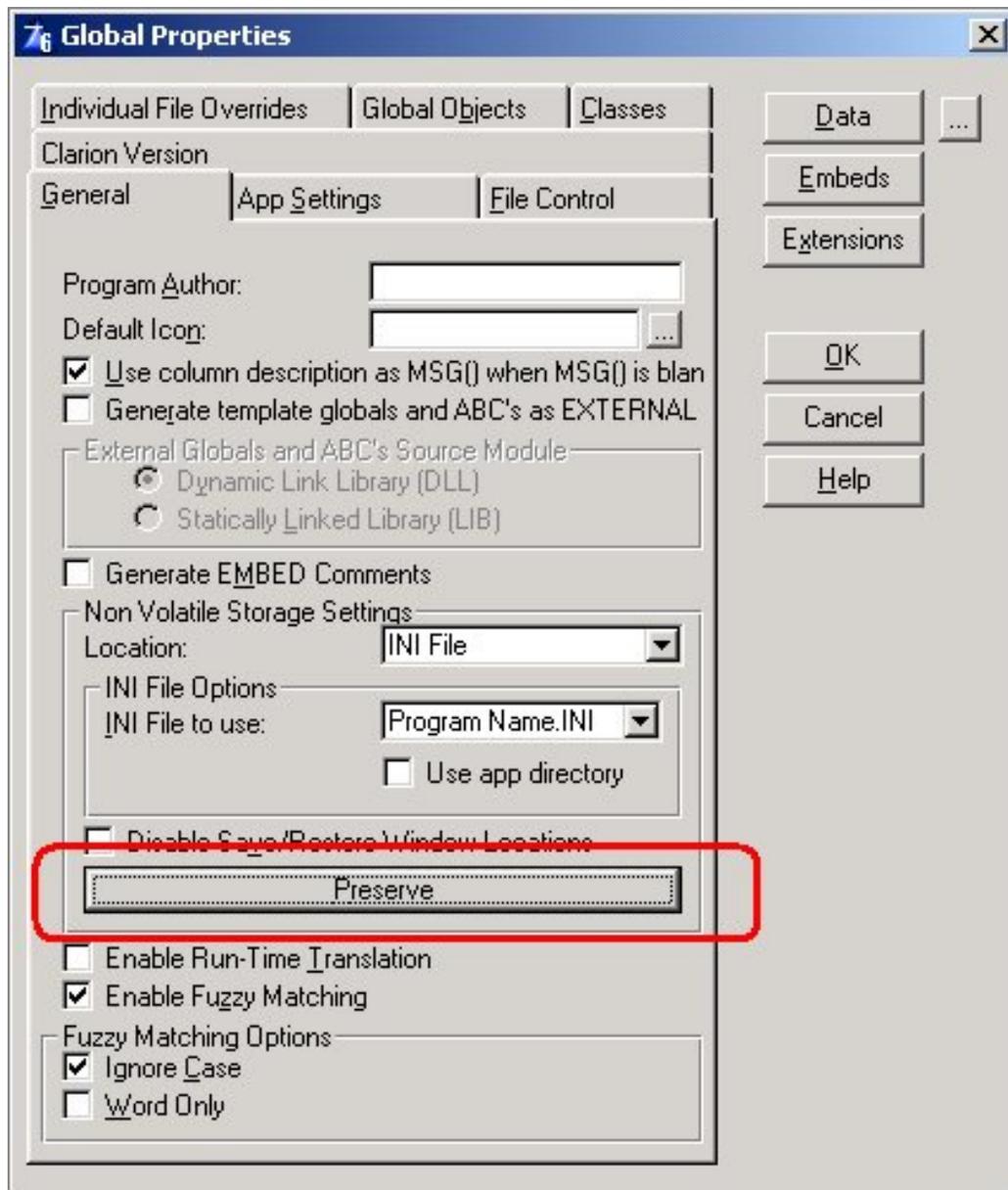
or

```
Access:VENDORS.SetName(GLO:DataPath & GLO:Vendors)
```

ensures that every NAME variable is set correctly, containing the fully qualified DOS file specification.

If you are not using a CString for the path variable, you *must* CLIP the path variable. Also, if you store the global path variable in an INI file (or in the Registry), the application's built-in ability to

retrieve and store INI file values cancels the need to manually retrieve the path value. See Figure 6.



**Figure 6. Global access point for INI variables**

Using variable file names is very, very flexible. It allows files to be anywhere visible to the user's computer. It allows multiple files to share a common structure and to be affected by common code.

It also requires that every file used in an application have its variable set, using one of the two statements shown above.

This could mean dozens or hundreds of such statements. And, *you* have to type them. One typo ... if you're lucky, you'll get a compiler error and have a chance to correct the mistake. With less luck, two variables will share a common NAME but not a common structure. Errorcode 47 is the inevitable result. And tracking this down can be a very frustrating experience.

## Introducing PROP:DataPath

Clarion 6 introduces a new system property that eliminates all these hassles and makes working with the NAME attribute simply irrelevant.

The Clarion 6 *Migrating from Prior Versions* document says:

One of the nice new features of this release is a new system property:

```
SYSTEM{PROP:DataPath}
```

With this, you can set your data file names in the dictionary to have no path in them, and then set the data path once in your program start up. From there, each file will inherit the common data path.

In other words, if I change the code to get the data path to:

```
GLO:TMP = 'datapath.fil'
Access:TMPASCII.Open( )
Access:TMPASCII.UseFile( )
Get(TMPASCII,1)
System{Prop:DataPath} = TMP:Data
```

all of the code to set the individual variables is unnecessary. In fact, declaring the variables is unnecessary.

At the end of this article you can find an application using SYSTEM{Prop:DataPath} shows that this is exactly correct (you will need to create two directories, DATA and MOREDATA, below the app or you can modify the code).

In the example program, no NAME variables are used at all. The application has two browses. One looks for its data files in .\DATA; the other looks in .\MOREDATA. The application behaves exactly like NAME variables *are* present and are set to .\DATA\VENDORS.TPS and .\MOREDATA\VENDORS.TPS. In fact, if you open the browse that looks into MOREDATA then open that browse that ought to look at DATA, you will in fact get the data from MOREDATA, just as if GLO:Vendors had been initialized to .\MOREDATA\VENDORS.TPS.

## Summary

The on-line help describes PROP:DataPath as follows:

A read/write SYSTEM property that can be used to set the default directory for data files.

All files with unqualified file names (e.g., those files with no NAME attribute or a NAME with only a relative path) will be looked for in the directory specified by SYSTEM{PROP:DataPath}.

SYSTEM{PROP:DataPath} defaults to the directory the application starts in. This will save developers from having to do startup code like the following:

```
GETINI(datadirectory)
file1name = datadirectory & 'file1'
file2name = datadirectory & 'file2'
file3name = datadirectory & 'file3'
```

The short version of this is that you don't have to bother with NAME, one global variable per file or with setting those variables any more. You don't have to bother with completing the Full Pathname prompt in the dictionary and you don't have to declare any of these variables any more.

[Download the source](#)

---

*"hp" in fact prefers Hewlett-Packard printers but will use whatever is available. Born in New York City, hp is a self-taught Clarion developer doing a substantial amount of work for hospital gift shops.*

## Reader Comments

[Add a comment](#)

**Hello Henry, Great article. My only questions is how...**

**Hey Glenn, You could specify it when you prime the...**

**Hello Steven, I was just curious where the suffix was...**

**Yes, the driver supplies the extension if you don't supply...**

**You can do the same thing in Clarion 5.5 and before simply...**

## Clarion Magazine



# Moving Applications to Oracle: RI And AutoNumbering Part 3

by Jon Waterhouse

Published 2003-12-05

[Last week](#) I finished up my discussion of autonumbering in Oracle. This week it's time to look at another important issue in Oracle (and other) databases: transactions.

Transactions are units of work that take a database from one consistent state to another. Whether your back end is an Oracle database or Clarion TPS files, one thing you want to prevent happening is having a database end up with half a transaction in it. The classic example is a transfer between two bank accounts. One part of the transaction is to subtract the money from the first account; the second part is to add it to the second account. Either both these actions should take place, or neither of them. The second thing that is important in transactions is how they incorporate changes made concurrently by other users. For example, if you are trying to calculate and store bank balances, and there are transactions happening at the same time at banks and ATMs across the country, which information is incorporated and which is discarded?

Clarion has three commands that are designed to set up transactions, which are translated into different commands for each back end. The Clarion commands are LOGOUT, COMMIT and ROLLBACK. In Clarion the set of actions carried out between the LOGOUT and the COMMIT is the transaction.

In Oracle, everything between one commit and the next commit is a transaction; there is no equivalent of a LOGOUT statement (this is not true in several other SQL databases). There are also some activities (e.g. involving the DDL, such as creating tables and altering table structures) that implicitly force a commit. Rollbacks occur either when errors are encountered or a ROLL BACK statement is encountered.

There are two other statements that control transactions in Oracle; SAVEPOINT allows you to break a transaction into parts so that if a rollback happens it the database is rolled back only to the savepoint (rather than the last commit), and PRAGMA AUTONOMOUS\_TRANSACTION is a stored procedure directive that allows transactions within transactions. One use of this is to allow a logging procedure to document what someone attempted to do even if the attempt

ultimately failed and was rolled back.

The points in a Clarion program when transactions are most important are when inserting, deleting and modifying data in the database. Client side auto-numbering on insert is a potential additional source of data modification, separate from the "real" insert. Auto-numbering managed by Clarion is not a good idea in Oracle databases (see [Part 2](#)). In the browse-form paradigm inserts, changes and deletes mostly happen when a form is completed.

The other major place where data modification takes place is in batch transactions carried out in process procedures. If you are writing with an Oracle back end in mind you will likely never use process procedures: most processes will translate into a single SQL statement. In those instances where row-by-row processing is still required you will likely set up a cursor in a stored procedure to handle this rather than use a process in Clarion.

If the batch process is translated into a single SQL statement the transaction will include (at least) the entire batch process; if an error is encountered in one row, *none* of the changes will take place. The concept of putting a LOGOUT/COMMIT around an entire process is a bit of a foreign concept to a pure Clarion programmer. LOGOUT/COMMITs are sometimes inserted within process procedures to process (say) 100 records at a time, but this is normally done to improve performance. In Oracle, the rationale is mainly that it is a lot easier to deal with a transaction that fails entirely, than with one that alters half (or some of) the rows, and ignores others, leaving you to figure out into which category a particular row falls.

A second method of implementing the functionality of a batch procedure is to use a cursor. If you use a cursor you have a choice of whether to commit each row as it is processed, or to treat the entire batch as a transaction. There are still some difficult questions to work out. For example, suppose you are adding a new employee to your HR database. He has qualifications from some foreign university that you have to add to an existing list. When it finally comes time to save your new employee record in the database, you find you can't do it because you don't have his social insurance number (which is not allowed to be NULL). Should the qualifications you entered be considered part of the whole transaction and rolled back, or should the addition to the qualifications list be considered a completely separate transaction? By default, in Clarion it would be a separate transaction; in this case this is probably not a bad thing, in other cases you may want to consider some of these ancillary changes as part of one multi-faceted transaction.

The ABC templates that ship with Clarion try to deal with two transaction-related issues: concurrency control and relational integrity. I began a discussion of relational integrity in [Part 1](#). The following section will discuss concurrency issues. Understanding how Clarion deals with basic concurrency issues will help understand how more complex transactions can be managed.

## Concurrency control

Concurrency control has to deal with two users making changes to the same record. Oracle will happily give any number of users a copy of a particular table row to look at. If more than one of those concurrent users decides she wants to make changes to the record and save those changes

back to the database, you have a problem. There are two ways to deal with this. One is for the program to lock rows where it is likely that the user will update it (e.g. when the user opens a form) so that someone else who comes along is told the record is in use if they also try to update the same row. The second is to check just before changes are saved to make sure that someone else has not changed the row while we have been preparing our changes. I guess there is also a third way, which is let people save whatever they want without any checks, but users don't react very well to finding things unchanged that they *know* they have changed (but someone else wrote over their changes afterwards in changing something else in the row at the same time).

The two solutions described above are known as pessimistic and optimistic locking respectively. Pessimistic concurrency says it's fairly likely someone else will muck with my data before I'm ready to save, so I should lock it now; optimistic concurrency says it's unlikely anyone else will have done anything to my record, but I'll check just to make sure, and will accept the risk that I'll have to re-enter my changes if that happens..

The first method (pessimistic) is accomplished in Oracle by requesting data with a `FOR UPDATE` clause. Other people can still read the data, but they can't acquire the row `FOR UPDATE` themselves, nor can they update or delete the row without locking it first (which would be denied). In Clarion, to implement pessimistic locking, when the form opens the `SELECT` statement would have to select the row with the `FOR UPDATE` clause. You could accomplish this by using the Clarion `SQLCallback` feature, but that isn't something I've tried.

To implement optimistic locking, just before saving changes, the application should re-read the data, check to see that the row still looks like it did when it was originally grabbed (giving you an error message if not), then save the changed row. Ideally, it should do this all in one operation so that there is no chance of a change happening in between. In SQL this can be accomplished by a statement like:

```
UPDATE mytable SET changedfields=:newvalues
WHERE changedfields=:oldvalues and allotherfields=originalvalues
```

This is what a standard Clarion application will do. Slightly less reliable, because there is the chance that a change will be made in between the check and making your change, is to carry out the check to make sure the row still looks the same first, and if that's OK, then make the change.

Up to this point I have been a bit slack in my use of the word *save*. Using flat files you think about things the following way: you grab the record off the disk into the record buffer in memory; you make the changes in memory; you save the changed buffer back to the disk. Game over. Anyone else comes along afterwards and reads from the disk, they see the changed record. With Oracle that is not true. [Next week](#) I'll explain why that's the case when I look at how Oracle commits transactions, and allows other users to read data that is in the process of being changed.

[Download the source](#)

*[Jon Waterhouse](#) has been using Clarion since the 2.1 days. His main work is as an economist, and he finds that Clarion is well-suited for applications which impose order on various sets of data. His projects include questionnaire data entry programs, classification software (assigning projects to groups), plus some more interesting scheduling applications. Jon has also used Clarion to link text information together, and is currently developing a program that will store linked snippets of WordPerfect documents and print custom documents composed of several of these snippets. He is currently working for the Newfoundland Government on a project to measure the performance of government employment programs.*

## 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

Reborn Free

**CLARION**  
*online*

## A Simple Versions And Annotations Template

by Ronald van Raaphorst

Published 2003-12-05

I know I work hard and do make progress every day. But I often do not switch from Clarion to another application to write down all the fixes and improvements. I just keep forgetting to do it.

Still, documentation and version control is very important. For testing purposes, I sometimes copy some code to another app and change it. Then, months later, I need the same trick, but I don't exactly recall in which application I implemented it, let alone in which procedure.

What I really I wanted was some kind of annotation system, included in my Clarion app. I wanted to be able to write down changes in a procedure in the procedure itself, and print an overview of all annotations per procedure or application.

I had a vague idea that someone had done it in a template, but I didn't exactly recall where I had seen it. But, having become more and more confident with the template language, it seemed a nice challenge for a spare hour.

### The design

What I simply wanted, per procedure, was a list of all changes for that procedure, with an author (that's always me here at Compad), a date and a text field.

As you know there are several types of templates: procedure templates, control templates, and code templates and extension templates.

I had to opt for a global procedure extension template, because it can be added to every procedure. The template does not need to generate code (although it might be an option to include the annotations in the source, now that I think of it). It only needs to store the annotations.

Here's the source code for the first part of the template, showing the global extension:

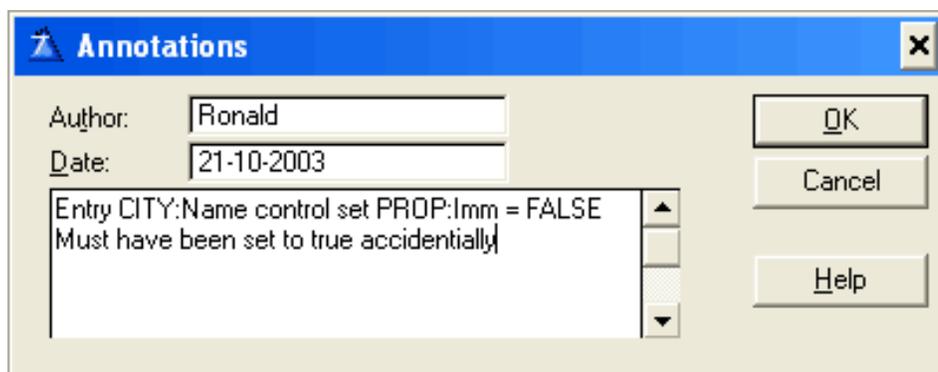
```
#! -----  
#TEMPLATE (cs_Annotations,   
    'Version and annotations extension'),FAMILY('ABC')  
#! -----
```

```

#! EXTENSION cs_Annotations
#EXTENSION(cs_Annotations,'Version and annotations extension'), ←
    PROCEDURE
#PREPARE
    #DECLARE(%csMaxInstance)
    #FOR(%csAnnotationInstance)
        #IF(%csAnnotationInstance > %csMaxInstance)
            #SET(%csMaxInstance, %csAnnotationInstance)
        #ENDIF
    #ENDFOR
#ENDPREPARE
#BUTTON('Version '&%csMaxInstance),AT(, ,180)
    #SHEET
        #TAB('Annotations')
        #DISPLAY('Current Version: '&%csMaxInstance)
        #BUTTON('Annotations'),MULTI(%csAnnotationInstance, ←
            FORMAT(%csAnnotationInstance,@N_3) &' ' ←
            &FORMAT(%csAnnotationAuthor,@S10)&%csAnnotationText),INLINE
        #PROMPT('Au&thor:', @S20),%csAnnotationAuthor,REQ,AT(50)
        #PROMPT('&Date:',@D6-),%csAnnotationDate,REQ, ←
            DEFAULT(TODAY()),AT(50)
        #PROMPT('A&notation:',TEXT),%csAnnotationText
    #ENDBUTTON
    #ENDTAB
    #ENDSHEET
#ENDBUTTON

```

Figure 1 shows the window displayed by this template.



**Figure 1. Fields used to store the annotation**

But how would I be able to have an overview of all changes? Then I remembered there's something called a utility template, which can be called directly, and I remembered there's an #OPEN statement, with which the template language can write to a file. Thus, why not use this utility template to create a tab delimited file, which can be read in MS Excel? In Excel you have the autofilter option, with which you quickly can filter non relevant things. And, you're able to print the overview in any way you want it. Figure 2 shows an example of such a file in Excel.

Application	Procedure	Version	Author	Date	Description
csupd	Update_City	v1	Ronald	21-10-2003	Must have been se

**Figure 2. Result loaded in MS Excel, autofilter on**

The utility template is as follows:

```

#! -----
#! UTILITY cs_ShowAnnotations
#! -----
#UTILITY(cs_ShowAnnotations,'Export annotations to a file')
#Sheet
  #Tab('Show Annotations')
    #DISPLAY('')
    #PROMPT ('File:',@s100),%workfile,DEFAULT(%Application&' '
    &FORMAT(TODAY(),@d6-)&'.ann'),AT(50,,140)
    #DISPLAY('')
    #DISPLAY('Show Annotations')
    #PROMPT('From:',@D6-),%csStartDate,AT(50)
    #PROMPT('Until:',@D6-),%csEndDate,DEFAULT(TODAY()),AT(50)
    #DISPLAY('(Leave blank for no filter)'),AT(50)
  #EndTab
#EndSheet
#Declare(%Outstring)
#Open(%workfile)
#SET(%OutString,'Application<9>Procedure<9>Version<9><9>
  Author<9>Date<9>Description')
%OutString
#FOR(%Procedure)
  #CONTEXT(%Procedure)
    #FOR(%ActiveTemplate),
      WHERE(%ActiveTemplate='cs_Annotations(cs_Annotations)')
    #FOR(%ActiveTemplateInstance)
      #CONTEXT(%Procedure, %ActiveTemplateInstance)
      #FOR(%csAnnotationInstance)
        #IF (%csStartDate=0 OR (%csStartDate > 0
          AND %csStartDate <= %csAnnotationDate))
          #IF (%csEndDate=0 OR (%csEndDate > 0
            AND %csEndDate >= %csAnnotationDate))
            #SET(%Outstring,%Application&'<9>'&%Procedure&'</pre>

```

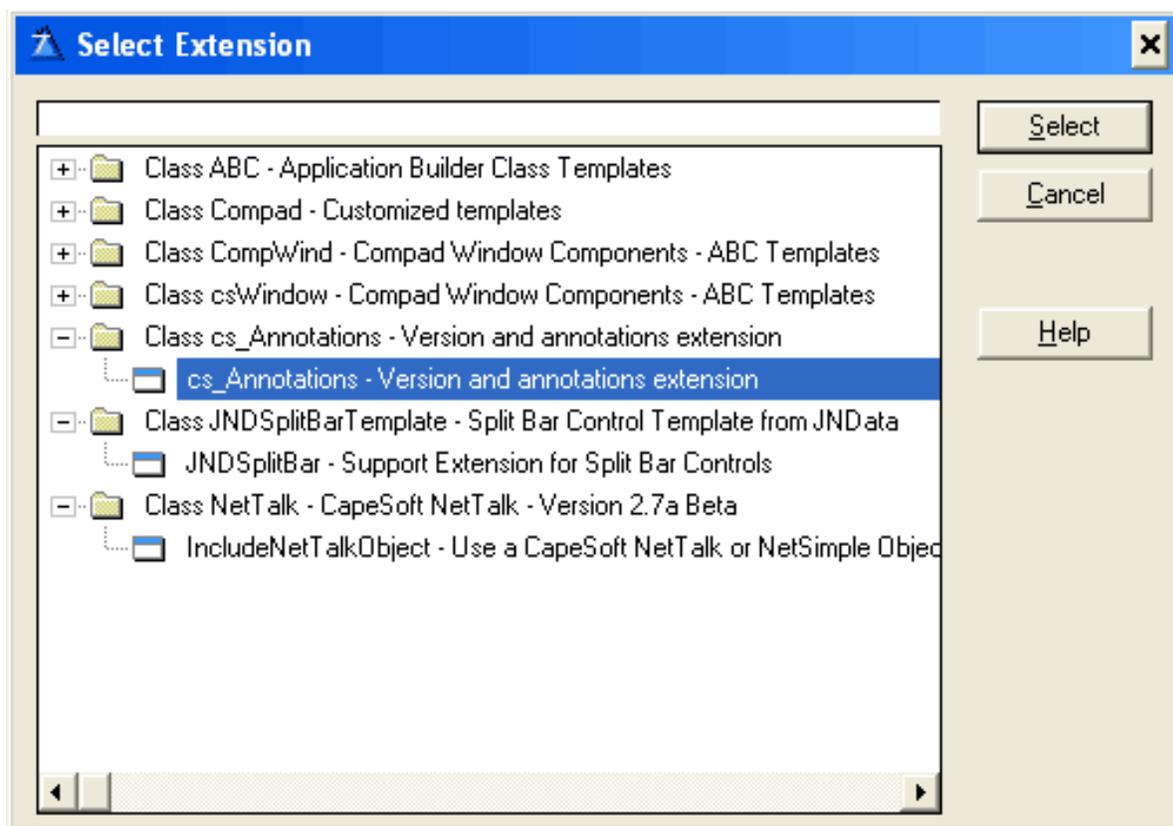
```

        <9>v'&%csAnnotationInstance&'<9>'&%csAnnotationAuthor
        &'<9>'&FORMAT(%csAnnotationDate,@d6-)&'<9>'&
        &%csAnnotationText&'")
%OutString
        #ENDIF
    #ENDIF
#ENDFOR
#ENDCONTEXT
#ENDFOR
#ENDFOR
#ENDFOR
#ENDCONTEXT
#ENDFOR
#Close(%Workfile)

```

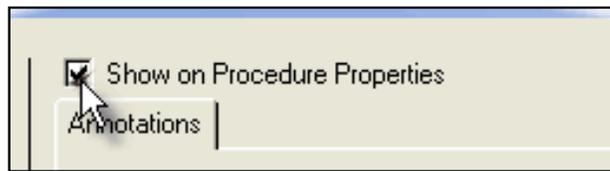
To use the template, you only need to download the source, and save the `cs_Annotations.tpl` file in the `\c55\3rdParty\Template` folder. Then register this template and you're ready to go.

First, open the application in which you want to use the template. Then, open a procedure, click on the Extensions button, and insert a new extension. Then, select the `cs_Annotations` extension, as in Figure 3.



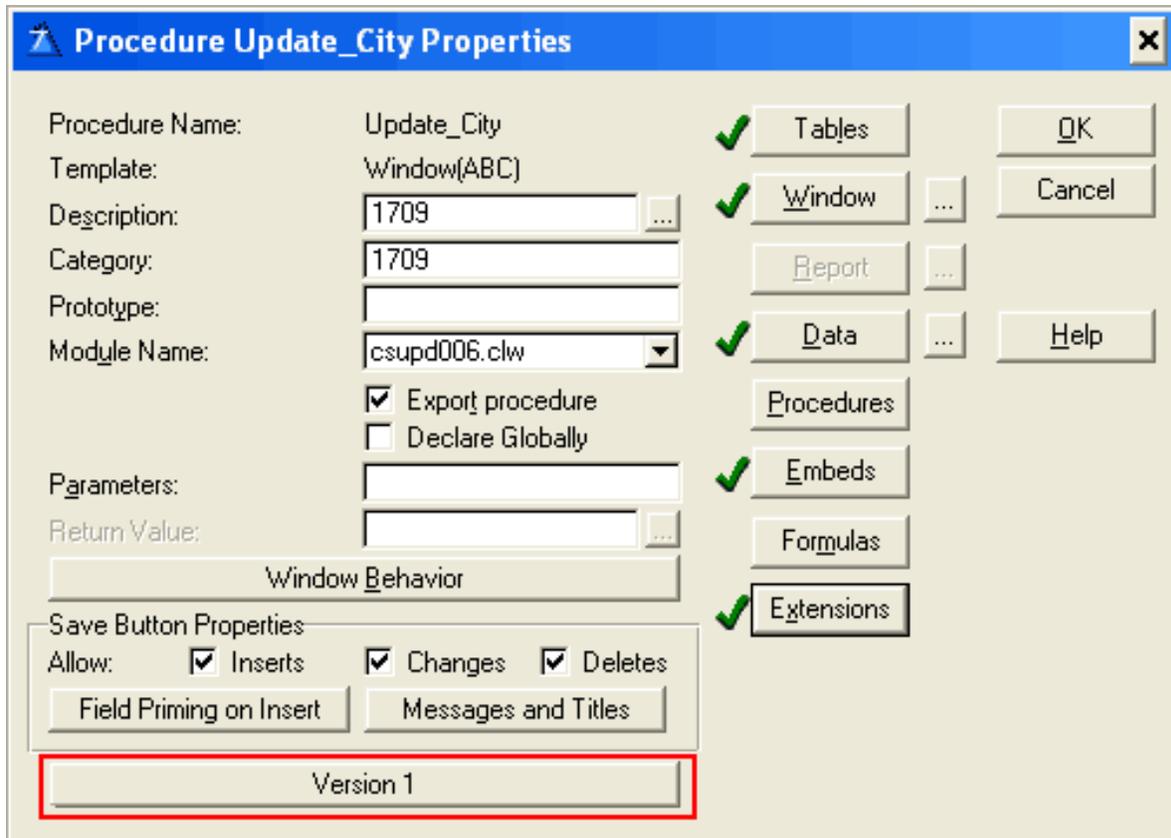
**Figure 3. Select the `cs_Annotations` extension**

I added a button on the properties window for that procedure, which shows the current version. This button is only visible if you check the Show on Procedure Properties check box (see Figure 4), and doesn't show up immediately; you have to close the procedure properties window and open it again.



**Figure 4. The Show on Procedure Properties check box**

To add a note, just click the insert button on the annotations tab and enter an annotation (see Figure 1). Once the Version button has been enabled for procedure properties (again, the first time you have to close/reopen the procedure window), you'll see something like Figure 5.

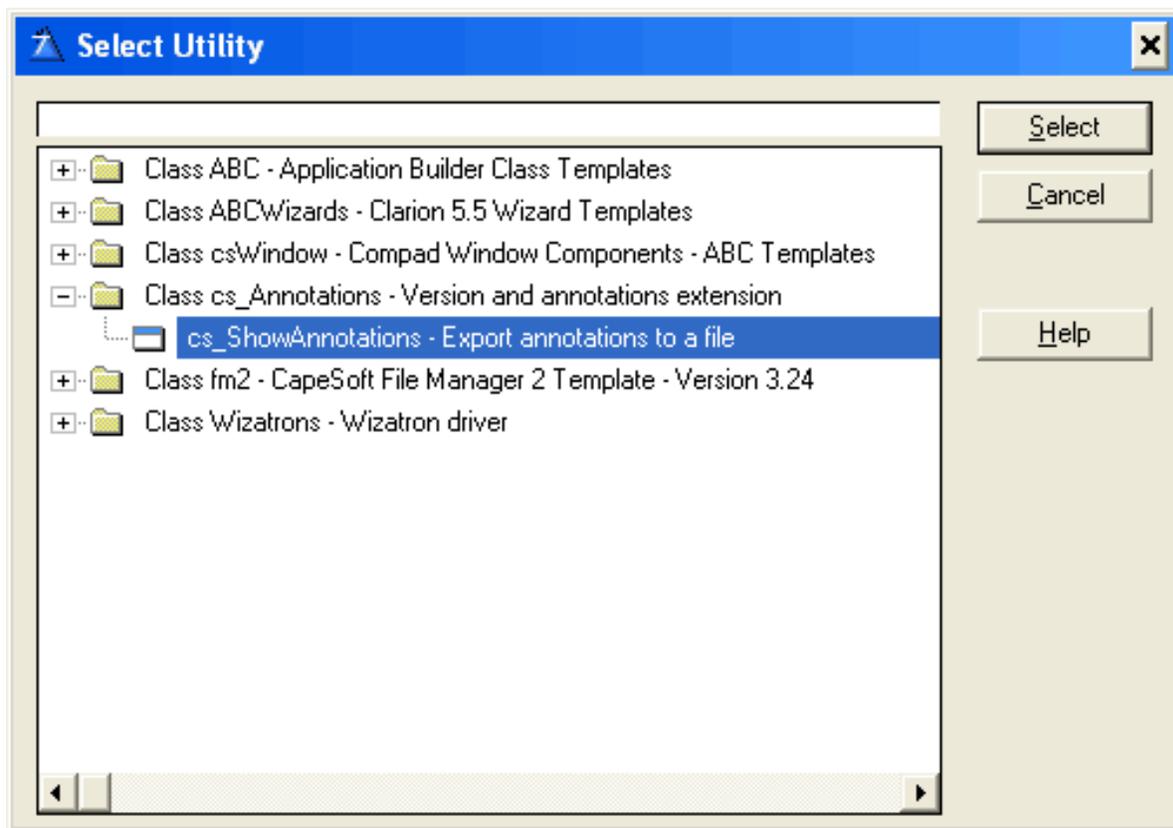


**Figure 5. Version (Number of annotations) showing in the procedure properties window**

This version number will be increased for every notation you make. By clicking the button, you can quickly add change notes.

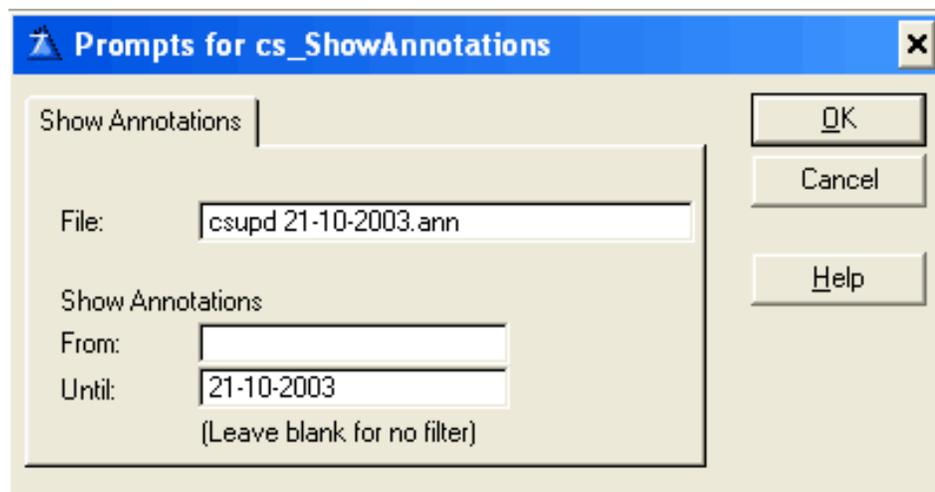
So, how do you get an overview of all changes in the application?

From the menu, select Application > Template Utility, or press CTRL-U. Then, select the cs\_Annotations utility template, as shown in Figure 6.



**Figure 6. Select cs\_ShowAnnotations utility template**

When you run the cs\_ShowAnnotations utility the window shown in Figure 7 appears.



**Figure 7. Select the output file and optionally enter filter dates**

The template automatically suggests a filename, composed of the application name and today's date. I choose to use the "ann" extension, because it's not commonly used as far as I know, and thus I can choose to set the Windows File Explorer to open this kind of file with MS Excel.

You optionally can enter dates as a filter, so you might restrict the overview to this month's changes or this week's changes. The file is created in your application map. Of course, you can enter a full path name such as "c:\my documents\changes\myfile.txt" if you wish.

That's all. Just open the file in MS Excel, and you have a nice overview of the application's change log.

[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](#)

**Hi: It would be nice if the annotation is on the embed...**

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

**Reborn Free**

**CLARION**  
*online*

## Clarion News

[Search the news archive](#)

### [thetingroup.com Clarion 6 Compatibility](#)

Clarion 6 versions of all thetingroup.com products are being tested now and will be posted on the web site by the first of the year.

*Posted Monday, December 29, 2003*

### [1st Logo Design Partnership Program](#)

1st Logo Design is establishing strategic partnerships with major service and technology providers. Sign up if you are interested in establishing a strategic relationship with 1st Logo Design and offering additional services for your customers.

*Posted Monday, December 29, 2003*

### [gNotes 3.0 Released](#)

Introductory pricing for gNotes 3.0 ends Jan 4, 2004. Full version \$79, reg. \$99, source code version \$169, reg. \$199. New in version 3.0: Search procedure; Status flags; New look and feel; Report built in; Available as source code. There is a \$39 upgrade fee if you do not qualify for a free upgrade.

*Posted Monday, December 29, 2003*

### [Jesus Moreno and Gitano Software Named Third-party Vendor of the Week](#)

Jesus Moreno and Gitano Software have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week. Jesus Moreno and Gitano Software are awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of gQ, their widely acclaimed querying tool for Clarion Developers.

*Posted Monday, December 29, 2003*

### [Clarion Developers Challenge Week 16 Winners](#)

First place for Week 16 of the Clarion Developers Challenge goes to Dean Burgess. Dean picked a solid 12 out of 16 to claim first place in this week's contest. Alejandro Contreras, David Potter, Bo Schmitz, Dave Bratovich, Michael Pollard and Jeff Nordgren all finished a close second with 10 correct picks. For finishing first in this week's Clarion Developers Challenge, Dean will be receiving a copy of Setupbuilder donated by Friedrich Linder and Lindersoft. Dean will also be receiving a copy of Product Scope 32 PRO, single user, spreadsheet license from David Troxell and Encourager Software. The winner of this week's

special "Encourager" Award goes to Sebastian Streiger. Sebastian won the Encourager Award this week after winning top place last week. For the overall standings, Jerry Davis and Dean Burgess are in a tie for first place with a total score of 133. Jason Johnson, Dave Beggs, Larry Craddock, Tom Ferguson and David Potter are all just a step behind, with a number of other players still able to overtake the leaders going into the last week.

*Posted Monday, December 29, 2003*

### **[Simssoft C6 Compatibility](#)**

All Simssoft templates are now C6 compatible. Free upgrades as well as a comprehensive demo are available.

*Posted Monday, December 29, 2003*

### **[CapeSoft Tip of the Week](#)**

Two tips from CapeSoft, one on opening templates as Clarion source, and the other code generation.

*Posted Monday, December 29, 2003*

### **[BST 2.5](#)**

BST 2.5 has 20 new features. The price has been raised to \$299, upgrade from 2.3 is \$39. In a Nutshell, added Tooltips, drag and drop, doubleclick, and popup menu to schedulers. Requires Autoinc SysId and Key on Appt and reminder files for new features. These have been included in the txd files supplied with templates if you used them. The New features are not supported in C5 (IDE limit-Tooltips not supported).

*Posted Monday, December 29, 2003*

### **[Microsoft Demos and Downloads](#)**

Lots of interesting downloads here. Thanks to Brian Staff for the link.

*Posted Monday, December 29, 2003*

### **[DeveloperPLUS/Lodestar Software Holiday Schedule](#)**

DeveloperPLUS and Lodestar Software will be closed 20-Dec-2003 through 4-Jan-2004. What does that mean? Well, it means no business phones answered and maybe a slight delay in email response.

*Posted Monday, December 29, 2003*

### **[Clarion Handy Tools O8A1.4](#)**

A new CHT build, number O8A1.4, is now available. This update has been provided as an install, rather than as a set of Live-Update files because of the number of files involved. Live-Update is really intended for smaller, incremental changes in code, template and app files.

*Posted Monday, December 29, 2003*

### **[Microsoft ISV Program](#)**

Michael Lawson points out that Microsoft is offering entry into its ISV program for \$350, which includes five developer licenses to MSDN Universal.

*Posted Monday, December 29, 2003*

### **[Lost Editor Window Finder Utility](#)**

Carl Barnes has created a free "Clarion Editor Lost & Found" utility which should always be able to get focus back to your Clarion editor if it goes AWOL.

*Posted Wednesday, December 17, 2003*

### [Fenix Newsletter #1](#)

The first Fenix Newsletter is now available. Read the latest in the world of Fenix.

*Posted Wednesday, December 17, 2003*

### [xXPpopup 1.1](#)

SealSoft's xXPpopup v1.1 is now available. New in this version: XP Style popup menu now supported for Entry and Text fields; Small menu item bug fix.

*Posted Wednesday, December 17, 2003*

### [MAV Direct ODBC Versions 005 And 002](#)

MAV Direct ODBC new versions (005 and 002) have been released. Library version 0.05 includes bug fixes and has been tested with Firebird. Library version 0.04 interim adds class methods. Price is \$99 during beta, \$249 at gold release. Template version 0.02 interim has updated templates, doc, example app and a benchmark application. Price is \$99 during beta testing, \$149 at gold release.

*Posted Wednesday, December 17, 2003*

### [Informatic Consulting Named Third-party Vendor of the Week](#)

Informatic.Consulting has been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week! Jens Eden and Informatic.Consulting are awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of their TPS.Repair templates.

*Posted Wednesday, December 17, 2003*

### [Clarion Developers Challenge Week 15 Winners](#)

First place for Week 15 of the Clarion Developers Challenge goes to Sebastian Streiger! Sebastian picked a solid 12 out of 16 to claim first place in this week's contest! Jeff Nordgren and Alejandro Contreras finished a close second with 11 correct picks. For finishing first in this week's Clarion Developers Challenge, Sabastian will be receiving a copy of the TPS.Repair Templates from Jens Eden and Informatic.Conulting. The winner of this week's special "Encourager" Award goes to Mihai Palade. Mihai managed to edge out four other players with 6 correct picks to take home this week's Encourager Award. The "Encourager" Award, a copy of Product Scope 32 PRO, single user, spreadsheet license, is presented by David Troxell and Encourager Software to the player that finishes the week in last place.

*Posted Wednesday, December 17, 2003*

### [IceTips December 2003 Newsletter](#)

The IceTips December newsletter has news and tips from IceTips, and even some pictures from Arnor and Sue's recent trip to Iceland.

*Posted Wednesday, December 17, 2003*

### [ClarionNET Available From ClarionShop](#)

December 11th, 2003 marks the end of the three year exclusive distribution agreement between ClarioNET Solutions and SoftVelocity. ClarioNET Solutions will now handle sales and new user support for the ClarioNET product line. ClarioNET Solutions will continue to deliver updates and upgrades to ClarioNET. The next release will be Version 1.4, a free update to the Clarion 5.x based edition. Included are higher levels of encryption and internal polishing to refine the product. Documentation will be revised and supplied as PDF, HLP and HTML. The next "upgrade" will be ClarioNET for Clarion 6. This will be a fee based upgrade. Contact support@clarionet.com for more details. The parent company of ClarioNET Solutions is ENERCALC, INC., an engineering software developer in business since 1983.

*Posted Wednesday, December 17, 2003*

### **[Clarion Developers Challenge Week 14 Winner](#)**

First place for Week 14 of the Clarion Developers Challenge goes to Tom Ferguson. Tom picked an amazing 14 out of 16 to claim first place in this week's contest. Jerry Davis, Rick Martin and Kevin McHugh finished tied for second with 12 correct picks. For finishing first in this week's Clarion Developers Challenge, Tom will be receiving a copy of the MessageBox from Bruce Johnson and the other fine folks at Capesoft Software. The winner of this week's special "Encourager" Award goes to Bo Schmitz.

*Posted Wednesday, December 17, 2003*

### **[#cw-talk Back Up](#)**

Ron Schofield reports that #cw-talk is back up and running.

*Posted Wednesday, December 17, 2003*

### **[xFunction Library 2.0](#)**

SealSoft's free xFunction Library v2.0 is now available. This version is all Clarion code, no black boxes.

*Posted Wednesday, December 17, 2003*

### **[See Pictures Of Our New Clarion Book](#)**

The first proof copy of our new Clarion book is back from the printers - see pictures now!

*Posted Friday, December 12, 2003*

### **[Tips & Techniques Book On Sale December 15](#)**

Clarion Magazine's new Tips & Techniques book will go on sale December 15, 2003.

*Posted Monday, December 08, 2003*

### **[EasyMultiTag 2.06](#)**

EasyMultiTag 2.06 is now available. Changes include: Fixed problem with the Browse Queue formatting in some cases; Fixed GPF when using without active key.

*Posted Monday, December 08, 2003*

### **[SetupBuilder 4.03 C6 Compatible](#)**

Lindersoft has released the SetupBuilder 4.03, which is now compatible with C6. Editions available include: Clarion Edition (C55EE users only); Standard Edition; Web Edition. This will be the last 4.x release. SetupBuilder 5.0 beta is nearly code complete. If you're planning to buy SetupBuilder, then download SetupBuilder 4.03 now and order it before the price goes up.

Buy SetupBuilder 4 Standard or Web Edition and get SetupBuilder 5 Standard (\$249 value) or Professional (\$399 value) free.

*Posted Monday, December 08, 2003*

### **[Firebird ODBC Driver](#)**

Kelvin Chua reports that he has tested the latest free Open Source Firebird ODBC driver with Firebird 1.5 RC 7 and it seems to be that everything is working as expected. Speed had been as good as the EasySoft ODBC or GEMINI ODBC drivers.

*Posted Monday, December 08, 2003*

### **[kbalertz.com Indexes MS KB](#)**

Carl Barnes points out this site which indexes and categorizes Microsoft's knowledgebase in some very useful ways. You can also sign up to receive email notification of new articles.

*Posted Monday, December 08, 2003*

### **[ClassViewer 6.0 Version 2003.12.03](#)**

All registered users of ClassViewer have been notified via email of the availability of the latest version. If you did not receive your notification, please contact Randy Rogers.

*Posted Monday, December 08, 2003*

### **[AFE for C6Gold\(ABC\)](#)**

The AFE6(ABC) install for C6 Gold is available for download. This install requires the password & serial number for your AFE v2.5 server upgrade.

*Posted Monday, December 08, 2003*

### **[Capesoft Software Named Third-party Vendor Of The Week](#)**

Capesoft Software has been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week. Capesoft is awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of Capesoft's MessageBox.

*Posted Monday, December 08, 2003*

### **[BIT, BOT, BST Updated For C6](#)**

The Big Image Tamer, Big Scheduler Tamer, and Big Outlook Tamer templates have been updated for C6, Legacy and ABC.

*Posted Monday, December 08, 2003*

### **[ADDA Release Candidate](#)**

ADDA (Advanced Data Dictionary Architect) is now available for download. ADDA is a multipurpose toolkit for designing, creating and maintaining the database layer throughout the entire application lifecycle. It is especially useful to those developers who use Clarion in conjunction with .NET tools. Highlights include : Database primary planning; Keep track of most recent changes; Using pool of columns for common use to speed up development time; CASE like database display; End user can add customized tables and columns; Instant automatic database creation; Fully automatic upgrade of customer's database; Open file format (XML); Reverse-engineer existing databases; Generate SQL creation scripts; Print data dictionary; Optional automatic Clarion date and time conversion to datetime format; Add-in for MS Visual Studio displaying the contents of data dictionary in Clarion-like fashion.

*Posted Monday, December 08, 2003*

### **BigTamer Christmas Special**

The BigTamer Template and Utility Christmas Special is on until December 24, 2003, midnight Central Standard time. Purchase any BigTamer template at the regular price, and request up to two more templates free. The sum of the free template(s) price(s) must be equal to or less than the purchased template price.

*Posted Monday, December 08, 2003*

### **1st Logo Design Holiday Sale**

1st Logo Design is having a holiday special which includes 134 high resolution images (for toolbars, splash screens, web sites, and more); 345 icons; 1512 buttons; free copy of Accent, a handy Windows application that makes you more productive by organizing your notes, passwords, calendar, and life. Information that used to be buried on your hard drive is now at your fingertips; \$50 Gift Certificate for any custom work order of \$200 or more. Total value is \$548, buy now for \$199.

*Posted Monday, December 08, 2003*

### **DET For C6**

Dictionary Enhancement Templates (DET) v2.6 beta for Clarion 6, Clarion (Legacy) template chain is now shipping. All new features in the Clarion template chain are supported by DET, including ABC Classes, the new Browse List Format Manager, and the Form Entry Field enhancements.

*Posted Monday, December 08, 2003*

### **Clarion Developers Challenge Week 13 Winner**

First place for Week 13 of the Clarion Developers Challenge goes to Jerry Davis. Jerry pulled out the narrowest of victories over David Potter. Jerry and David both finished with 11 correct picks, so the contest was thrown into the Monday Night tiebreaker. Jerry chose a tiebreaker score of 37, while David chose a tiebreaker score of 46. The actual combined tiebreaker score was 41. That left Jerry with closest to the actual tiebreaker, with 4 to David's 5. For finishing first in this week's Clarion Developers Challenge, Jerry will be receiving a copy of the RPM: Report and Presentation Manager from Lee White and Lodestar Software. The winner of this week's special "Encourager" Award goes to Marc Walgren. For the overall standings, Bob Foreman has slightly increased his hold of the lead with a score of 107.

*Posted Monday, December 08, 2003*

### **BoxSoft C6 Templates Released**

BoxSoft has released C6 versions of the following templates: Super Import-Export, Super Browse, Super Field-Filler, Super Stuff (pre-documentation release), Super Invoice, Super Security, Super Limiter, and Super Passcode.

*Posted Monday, December 08, 2003*

### **Clarion API Database Updated**

A new version of the Clarion API database (freeware) is now available for download. The program is version 1.02, and the database is version 1.05. Please download the EXE, and do not upgrade via the program. In the future only data files will be updated. The database now

contains 1569 Prototypes; 305 Datatypes; 4223 Constants, and ready to use Clarion/VB/C++ and .NET examples.

*Posted Tuesday, December 02, 2003*

### **IMPEX 5.3**

Sterling Data's IMPEX is now compatible with Clarion versions 4 to 6.0.

*Posted Tuesday, December 02, 2003*

### **Lee White and Lodestar Software Named Third-party Vendor of the Week**

Lee White and Lodestar Software have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week! Lee White and Lodestar Software are awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of their RPM: Report & Presentation Manager. Lodestar has two flagship products, AFE: Automated Fax Engine, and RPM: Report and Presentation Manager. The Automated Fax Engine provides easy, feature rich support of fax send capabilities for your Clarion for Windows reports. RPM - Report & Presentation Manager provides you and your end users with total flexibility in report handling.

*Posted Tuesday, December 02, 2003*

### **IceTips Reporter For C6**

IceTips has made a new build of IceTips Reporter that is now Clarion 6 compatible.

*Posted Tuesday, December 02, 2003*

### **xPictureBrowse 2.3**

Sealsoft's xPictureBrowse 2.3 is now available. Changes include: Bug with crash program on procedure with ViewPicture control in DLL mode was fixed; Now xPictureBrowse require global extension template for export class definition; For a Multi-DLL program you must insert global template into main data app and into each app where you use xPictureBrowse class; xPicture browse source files renamed with short 8.3 name for compatibility with Clarion 5; New option to include/compile Windows XP manifest file into your program; Updated documentation and demo.

*Posted Tuesday, December 02, 2003*

### **SetupBuilder 5 Special Offer To Expire**

The next SetupBuilder installer generation is already looming on the horizon. After two years of development, SetupBuilder 5 beta, a complete rewrite of Version 4, will be available next month. The current special SetupBuilder 5 upgrade price will expire December 21, 2003.

Purchase or upgrade SetupBuilder 4 NOW and get SetupBuilder 5 Standard (\$249 value) or Professional (\$399 value) free.

*Posted Tuesday, December 02, 2003*

### **UK Clarion Meeting**

Russell Eggen and Mark Sarson would like to invite all UK Clarion users to London on the 12th December to meet up in the afternoon for a little get-together drink. Russell will be over in this country through most of December, and has asked to meet up with Clarion users.

Originally Russ was going to attend the December UKCUG meeting, but due to the changes going on at the moment, this had to be cancelled.

*Posted Tuesday, December 02, 2003*

### **BoxSoft C6 Compatibility**

BoxSoft has released Clarion 6 compatible versions of a number of products, including: Super QuickBooks-Export templates; Super Dialer templates; Super Tagging templates; Super Browse templates. For all purchase and upgrade issues (including passwords), please contact BoxSoft's distributor, Mitten Software. Their e-mail address is [Mitten@MittenSoftware.com](mailto:Mitten@MittenSoftware.com), and their phone numbers are (800) 825-5461 and (952) 745-4941.

*Posted Tuesday, December 02, 2003*

### **MAV Direct ODBC Template Version 0.01**

MAV direct ODBC Template version is an extended clone of the standard templates which works with the DB files (Control, Extension and Procedure Templates like Browse, Form, Process and Report) optimized to work with the MAV Direct ODBC library. \$99 during beta testing - \$149 when it goes Gold and free upgrade for beta testers. Requires MAV Direct ODBC library version. Clarion 5.0b, 5.5 and 6.0, ABC or Legacy, 32-bits only.

*Posted Tuesday, December 02, 2003*

### **Clarion Developers Challenge Week 12 Winner**

First place for Week 12 of the Clarion Developers Challenge goes to Michael Pollard. Michael scored an amazing 13 out of 15 to take first place in this week's contest. While Bob Foreman, Dave Sperling, and Jerry Davis finished a close second with 12 correct picks. Congratulations, Michael! For finishing first in this week's Clarion Developers Challenge, Michael will be receiving a copy of the Ace Icons Super Suite from Sue Pichotta and Ace Icons. Michael will also be receiving a copy of Product Scope 32 PRO, single user, spreadsheet license from David Troxell and Encourager Software.

*Posted Tuesday, December 02, 2003*

### **PowerXP Theme**

PowerXP Theme is a Global Extension Template that applies the correct XP-Theme when running your app under Windows XP. In addition, Groups, Radios and Checkboxes will be drawn with Windows XP-theme functions and they will become transparent. Applying PowerXP Theme to your application is done with less than five mouse clicks. No modification of your application is required.

*Posted Tuesday, December 02, 2003*

### **New Simshape and Simsoft Versions**

A new extension template has been added to the Simshape templates. Now you can turn the Simshape checkboxes into Option buttons with all the mouseover effects of the normal checkboxes. As well, a new template has been added so that the Simshape buttons now work with the Clarion(legacy) template chain. A further two templates have been added to the Simshape templates. Now you can use variables to hold the images for the checkboxes so that you can change the image group at runtime, and a corresponding Options extension has been added to allow you to use these checkboxes as options. There have also been a few improvements made recently (but not previously announced) to the Simsoft templates. These include new templates to allow variously sized keyboards to be called from a control button. The various calendars and diaries now allow the programmer to specify (or give his users the

option) whether the calendar should start with a Sunday or Monday (which apparently is the European convention). These upgrades are free to registered users. Eric Churton will confirm the C6 status of all his templates as soon as he gets his copy of C6. Upgrades if needed will be free to registered users. SimTabTree does apparently work fine with C6.

*Posted Tuesday, December 02, 2003*

### **[RADFusion Shopping Cart Goes Live](#)**

Russ Eggen reports that the RADFusion shopping cart test period is over, and the cart is live. If you have placed a test order, you will need to re-enter the order. Some of you will not have to re-enter your contact information. This is now the third site where you may place your order for the new Clarion book, "Programming Objects in Clarion.", besides DeveloperPlus and ClarionShop.

*Posted Tuesday, December 02, 2003*

### **[ClarionForge Unavailable Until Further Notice](#)**

ClarionForge will be unavailable while Ron Schofield builds the new server that will host ClarionForge, #cw-talk IRC and OpenClarion.

*Posted Tuesday, December 02, 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.

# Clarion Magazine



## Clarion 6 First Look: The IDE

by David Harms

Published 2003-12-11

This article is the fourth in a series in which I've provided an overview of the changes between Clarion 5.5 and Clarion 6. In the previous three installments I covered the changes in the [examples](#), the [source code](#), and the [templates](#). Last week I received my copy of the C6 Gold CD, so in this article I'll finish up that first look with an overview of the CD installation and the IDE changes.

As I was a participant in the Early Access program, I had the beta version still installed on my machine. SoftVelocity recommends a fresh install, so I uninstalled the Clarion 6 beta from Add/Remove Programs, via the Control Panel. Actually I was expecting to see a bunch of entries for the many beta patches, but there was only the one item. I successfully uninstalled the beta and rebooted. I then had a look at the C6 directories, which seemed to be pretty much intact. So I still had some cleanup to do.

I searched the registry, and didn't find anything to suggest that Clarion installs any Windows DLLs or other components. There is program configuration data on my machine under HKEY\_CURRENT\_USER\Software\SoftVelocity, none of which seemed worrisome. I decided to simply rename my old C6 directory, and I did a fresh install, which worked perfectly.

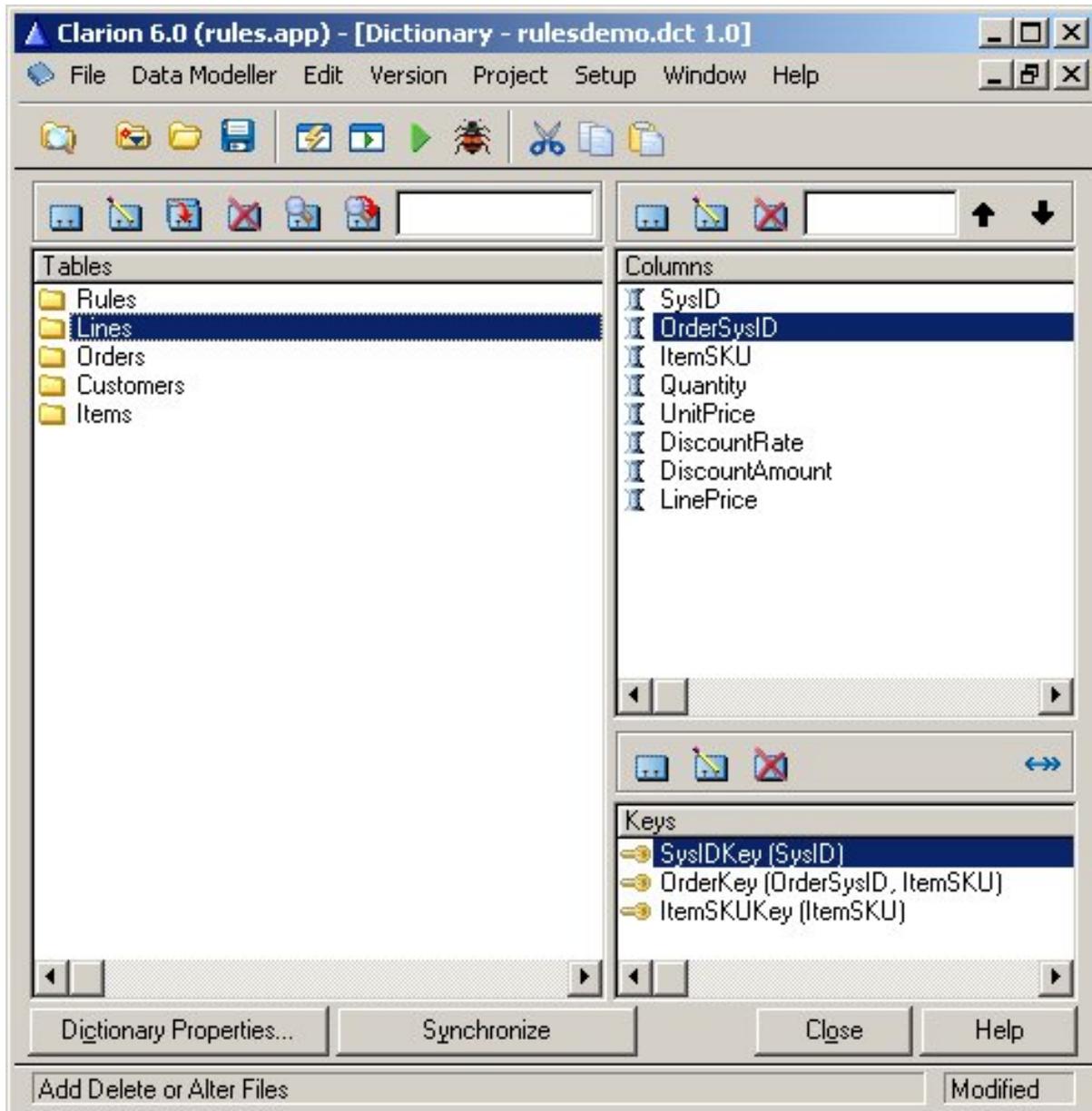
So what does Clarion 6 look like? In many ways, much like Clarion 5.x. There are no really dramatic changes to the IDE, but there are a whole lot of small to medium changes that improve the IDE's usability. And of course the many source and template improvements are also reflected in the options you have available to you when creating your applications.

### IDE-wide changes

One of the best modifications to the IDE is the addition of what SoftVelocity calls Smart Locators. In previous versions you could type a partial name (of, say, a procedure or table) and get the first match. Now you can press Ctrl-Enter repeatedly to go to the next match. It's a small change, but a very useful one. Also new is the Zoom Window feature. On just about any entry field, such as a procedure prototype or filter field, hit F10 to get a popup window that lets you edit the text in its entirety.

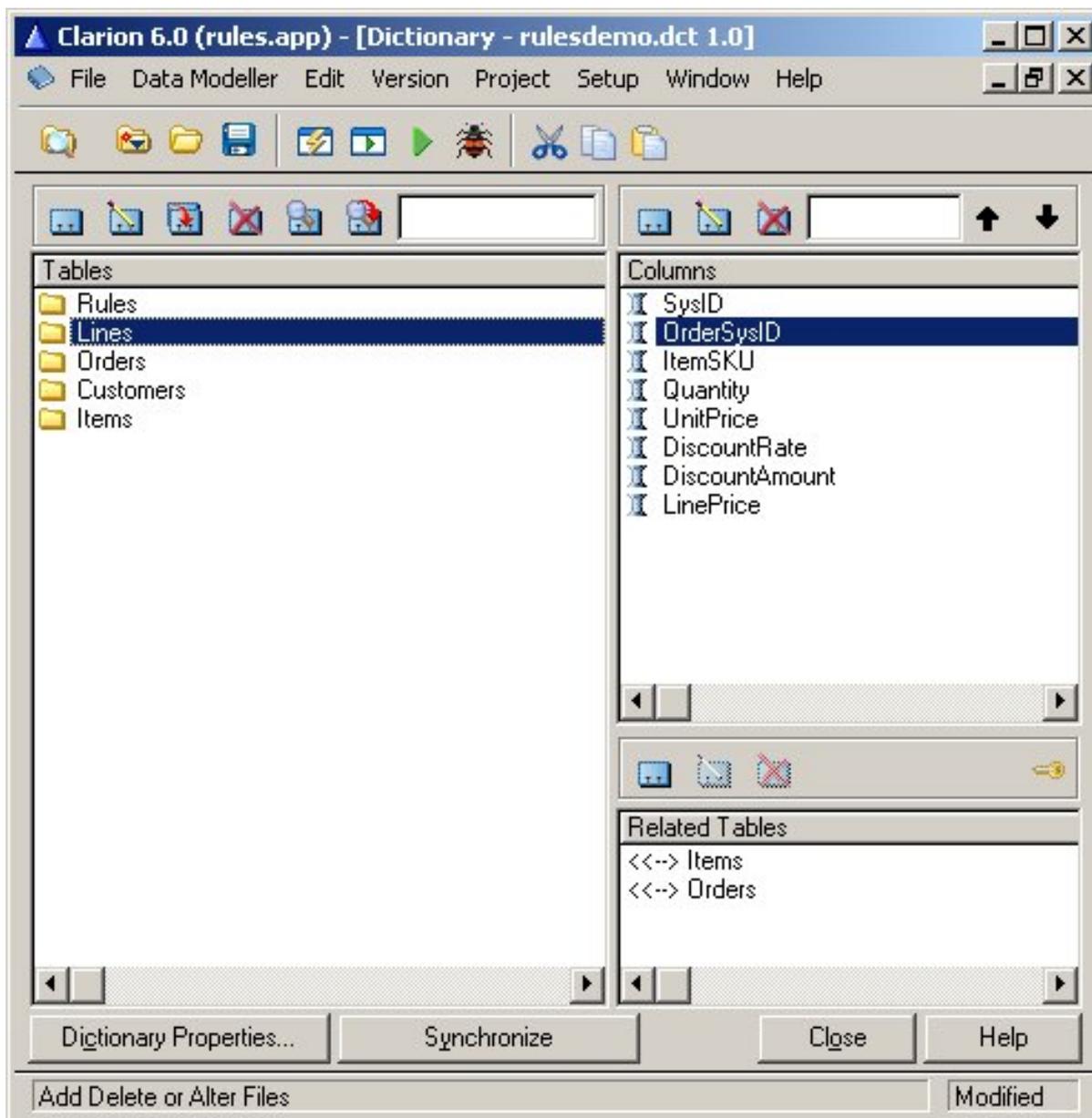
## The Dictionary Editor

The Dictionary Editor is where your applications begin, and it has a number of improvements to speed the creation and maintenance of your database files. Figure 1 shows the DE with the Business Rules example dictionary loaded.



**Figure 1. Dictionary main window, showing keys**

The first thing you'll notice is the three-panel view — tables on the left, columns on the right, and keys/relations on the lower right. I would still like to be able to drag the vertical separator the same as in the AppGen, but that's a minor quibble. By default the key list is shown, but just to the right of the key list's Add/Change/Delete icons is another button that lets you toggle the relations view. Figure 2 shows the DE with the relations displayed.



**Figure 2. Dictionary main window, showing relations**

Also new in the DE is an enhanced table options dialog. On the table properties window, click on the ellipsis button at the end of the Options entry field. The window you see depends on which driver you're using. For the TopSpeed driver, you'll see the one shown in Figure 3.

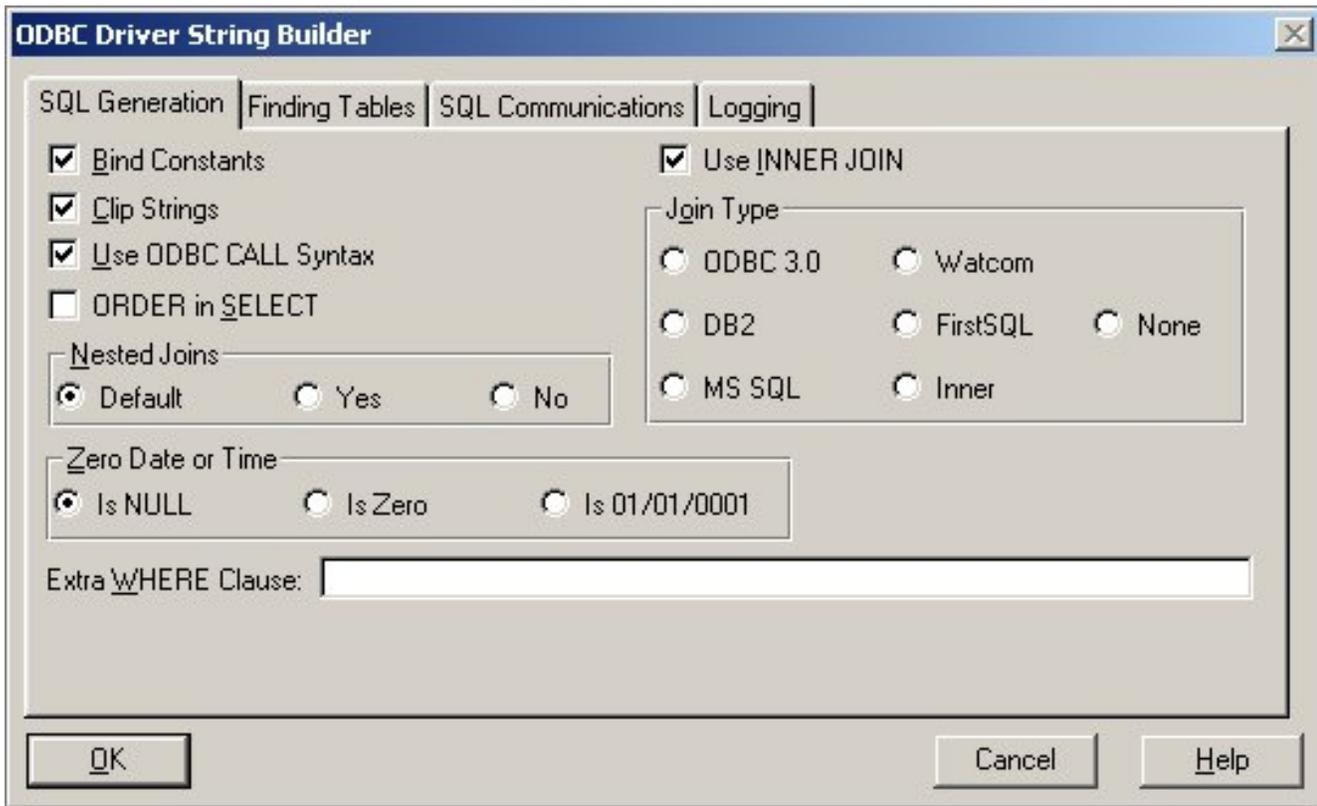


**Figure 3. TopSpeed driver options**

That's a pretty uncomplicated options window. If you specify a transaction control file, and disable decimal checking (which is enabled by default), the following string (or one like it) will be created for the Options field:

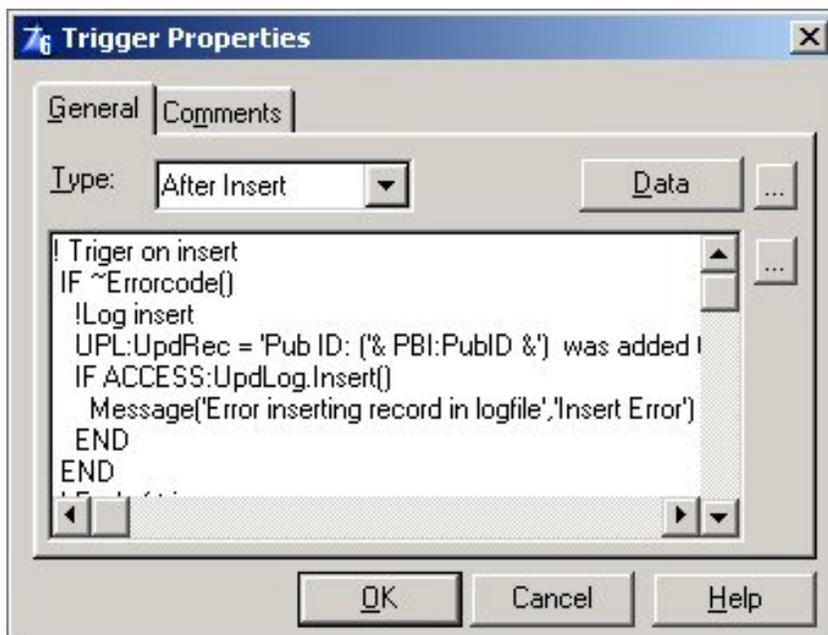
```
/DECIMALCHECK=OFF /TCF=my.tcf
```

Some drives have many more options. Figure 4 shows the first of four tabs of options for the ODBC driver.



**Figure 4. ODBC Driver options, SQL Generation tab**

As I noted in [Part 1](#) of this series, C6 supports client side triggers. These are blocks of code, which you create, and which execute always on an insert, change, or delete. The Triggers example app demonstrates this capability. To add a trigger, right click on the table and choose Triggers from the context menu. This brings up the Triggers window, in which you can add up to six triggers, i.e. Before and After Insert, Change, and Delete. Figure 5 shows an After Insert trigger



**Figure 5. An After Insert trigger.**

You can also declare data for triggers. Your code and data are generated as automatically called methods within the table's FileManager class (look for the *appnamebcn.clw* generated files).

One very significant change for the DE is that you can now synchronize the dictionary with any ODBC source. This is a huge improvement over C5.5 – I have to admit that I haven't used the synchronizer at all since its early days, but it looks like it's time for me to dive in. You can also now do multi-table import using ODBC.

## Application Generator

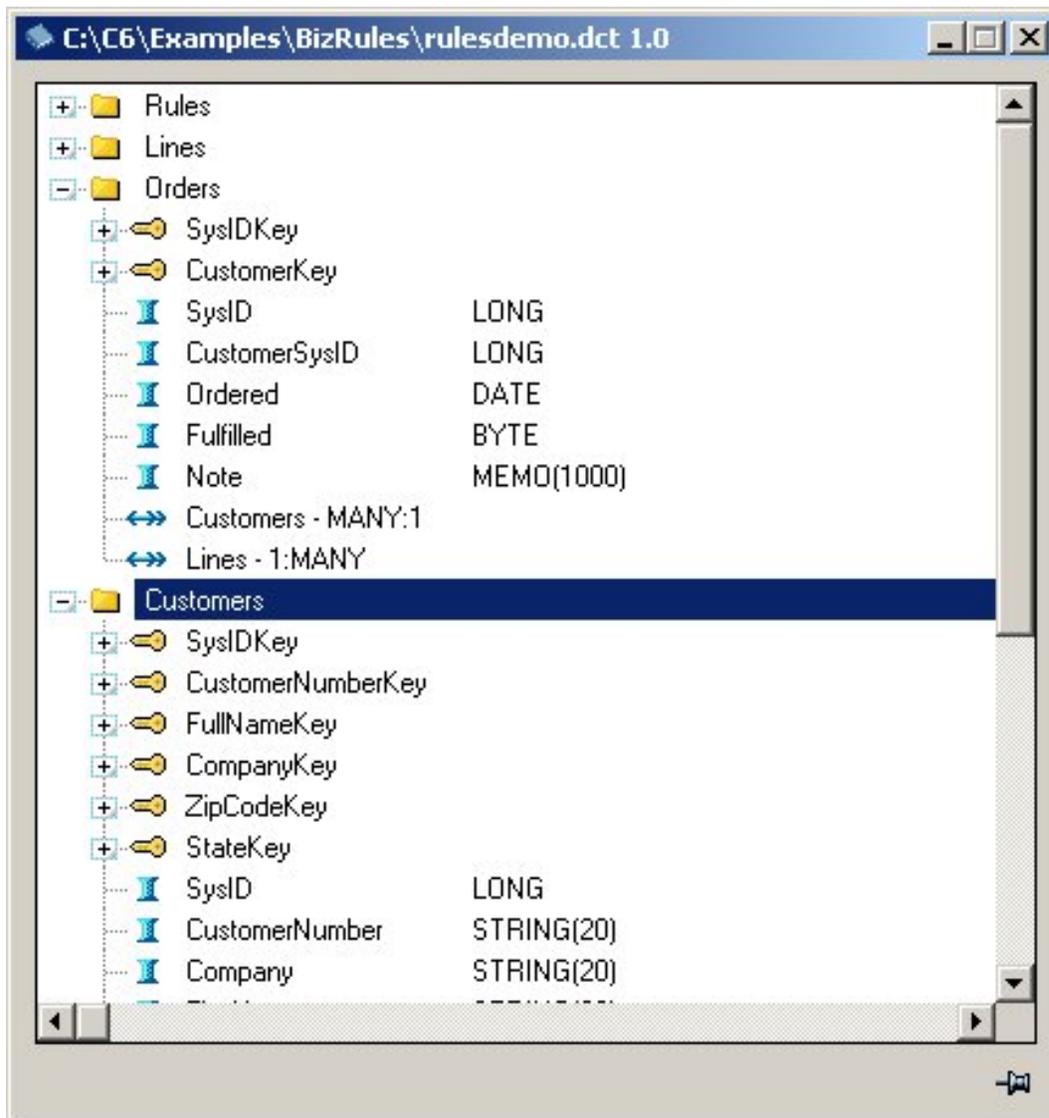
Most of the changes you'll notice in the AppGen are changes in code generation options/features. I've discussed most of these in previous installments on the source and templates.

The AppGen, like the DE, has had a bit of a facelift, with some new (and much-discussed during the beta period) icons (see Figure 6).

Notably, the famous "Blue Fart" icon is no more. In fact, instead of two icons for Make and Run, there are now three: Make, Make and Run, and Run. The Auto Make on Run menu option in C5.5 is no longer there in C6. If you're used to clicking icons to make your projects you'll have a fairly minor adjustment. If you're like me, and you always set Auto Make on Run on, then hit Alt-P-R to make and run an application, you'll be in for a surprise, because in C6 this key combination simply runs the application's EXE, if available. Creature of habit that I am, I still miss that auto-make feature.

There are also new icons to generate just the current module, and to browse the dictionary. These aren't new options, but sometimes it's good to be reminded they're there. For instance, I've never really used the View Dictionary option, yet it's quite handy. Bring up the dictionary, as shown in Figure 7, and you can click on the pushpin to lock the window to the forefront (and this will

really keep the window in the forefront, even if you switch to another application!). You can double-click (or right click) on items in the dictionary view to get read-only data which you can copy and paste.



**Figure 7. The dictionary viewer**

One new AppGen feature I do like is the ability to export a PRJ from an application. This can be particularly useful when you're debugging. Let's say you have a really nasty GPF somewhere in your application, and for one reason or another it's become almost impossible to track down. Really you're going to have to pare down your app until you can establish at least the procedure, or section of code, that's causing the problem. One way to do it is to make a copy of your app, start commenting out or deleting procedures, regenerating, compiling, and testing. But that wastes a lot of time, and is a bit dangerous since you may forget that you're working with a copy, or worse, you may comment out some code in a live app and forget you've done so.

A much better option is to create a PRJ that will let you compile the generated source. Then you simply hack up the generated source to your heart's content. If you hack too much, and have to start over, just regenerate the source code. When you've found the problem, again, go back and fix it in your app and continue on with development.

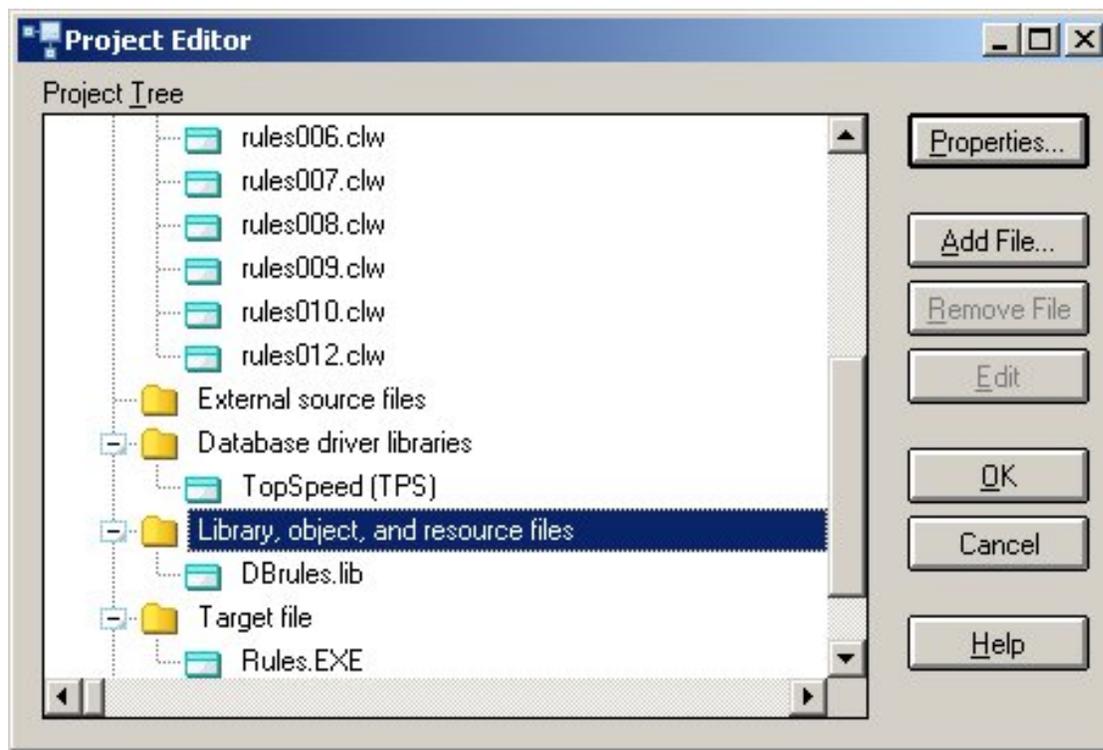
If you're not familiar with PRJ data, here's a quick primer. PRJ stands for Project, and the project data tells the compiler which files to compile, and the linker which files to link. You may think of hand coded apps as PRJs, and AppGen apps as APPs, but APPs also have PRJ data in them. It's just that until C6 it was harder to get the PRJ data out. You really had two options. One, which I've often used, is to open the APP in, say, Notepad, and search for the string #noedit. I would see a block of test something like this:

```
#noedit
#system win32
#model clarion dll
#pragma debug(vid=>full)
#pragma define(_ABCDllMode_=>0)
#pragma define(_ABCLinkMode_=>1)
#compile "TESTBC0.CLW" -- GENERATED
#compile "TESTBC.CLW" -- GENERATED
#compile "test.clw" /define(GENERATED=>on) -- GENERATED
#compile "test001.clw" /define(GENERATED=>on) -- GENERATED
#compile "test002.clw" /define(GENERATED=>on) -- GENERATED
#compile "test003.clw" /define(GENERATED=>on) -- GENERATED
#pragma link("C%V%ASC%X%%L%.LIB") -- GENERATED
#pragma link("C%V%ODB%X%%L%.LIB") -- GENERATED
#link "test.EXE"
```

The other way, pre-C6, is to export the app to a TXA and search for the project data there. Whichever way I got it, I would then copy everything down to the #link statement into another text file, and label it *appname.prj* or *appname.pr*. What's the difference? The Clarion IDE will open files ending in .pr with the text editor, and in .prj with the project editor. Other than that, there's no difference to the Clarion build system. I'd then close the app, use Project|Set to point to the PRJ, and start hacking up the generated source code.

In C6, getting a PRJ is simpler (and safer) than ever. Just open your app and choose File|Export Project File. Then close the app and load the project with Project|Set.

Another new project-related feature is the ability to include XP Manifest files and Version Info Script files. Choose Project|Edit and make sure that the Library, object, and resource files line is selected (Figure 8).



**Figure 8. The Project Editor**

Click on Add File, and the file dialog lists the file types shown in Figure 9.



**Figure 9. Library, object and resource file types**

Manifests are files with the same name as the EXE (or DLL) and with the extension .manifest. For instance, C6's libmaker application ships with the following manifest, called libmaker.manifest:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
  manifestVersion="1.0">
  <assemblyIdentity processorArchitecture="X86"
    version="6.0.0.0" type="win32"
    name="SoftVelocity.C60.LibMaker"/>
  <description>Clarion 6.0 Library Maker Utility</description>
  <dependency>
    <dependentAssembly>
```

```

    <assemblyIdentity
      type="win32"
      name="Microsoft.Windows.Common-Controls"
      version="6.0.0.0"
      publicKeyToken="6595b64144ccf1df"
      language="*"
      processorArchitecture="X86"/>
  </dependentAssembly>
</dependency>
</assembly>

```

Manifest files are only meaningful on XP systems; they will be ignored in earlier releases of Windows. And they can be placed in the same directory as their corresponding application, or linked into the application as C6 is capable of doing. The latter is usually preferable as it means one less file to ship, lose, or suffer unwanted modification by the user.

The main feature of the above manifest file, as far as users are concerned, is the `assemblyIdentify` section., which tells the operating system to use `ComCtrl32.dll` version 6.0 (which has the XP look and feel) when the application requests the creation of any control, such as a window, a button, or whatever.

Similarly, you can include Version Information Resource files in your application. Here's an example from the C6 help file:

```

LANGUAGE 0x419

1 VERSIONINFO
  FILEVERSION 1,0,0,1
  PRODUCTVERSION 1,0,0,1
  FILEFLAGSMASK 0x3fL
  FILEFLAGS 0
  FILEOS VOS__WINDOWS32
  FILETYPE VFT_APP
  FILESUBTYPE 0x0L
BEGIN
  BLOCK "StringFileInfo"
  BEGIN
    BLOCK "040904E4"
    BEGIN
      VALUE "CompanyName", "\0"
      VALUE "FileDescription", "This just a test\0"
      VALUE "FileVersion", "1, 0, 0, 1\0"
      VALUE "InternalName", "Version Info Script Example\0"

      VALUE "LegalCopyright", "Copyright (C) 2003\0"
      VALUE "LegalTrademarks", "\0"
      VALUE "OriginalFilename", "TEST\0"
    END
  END
END

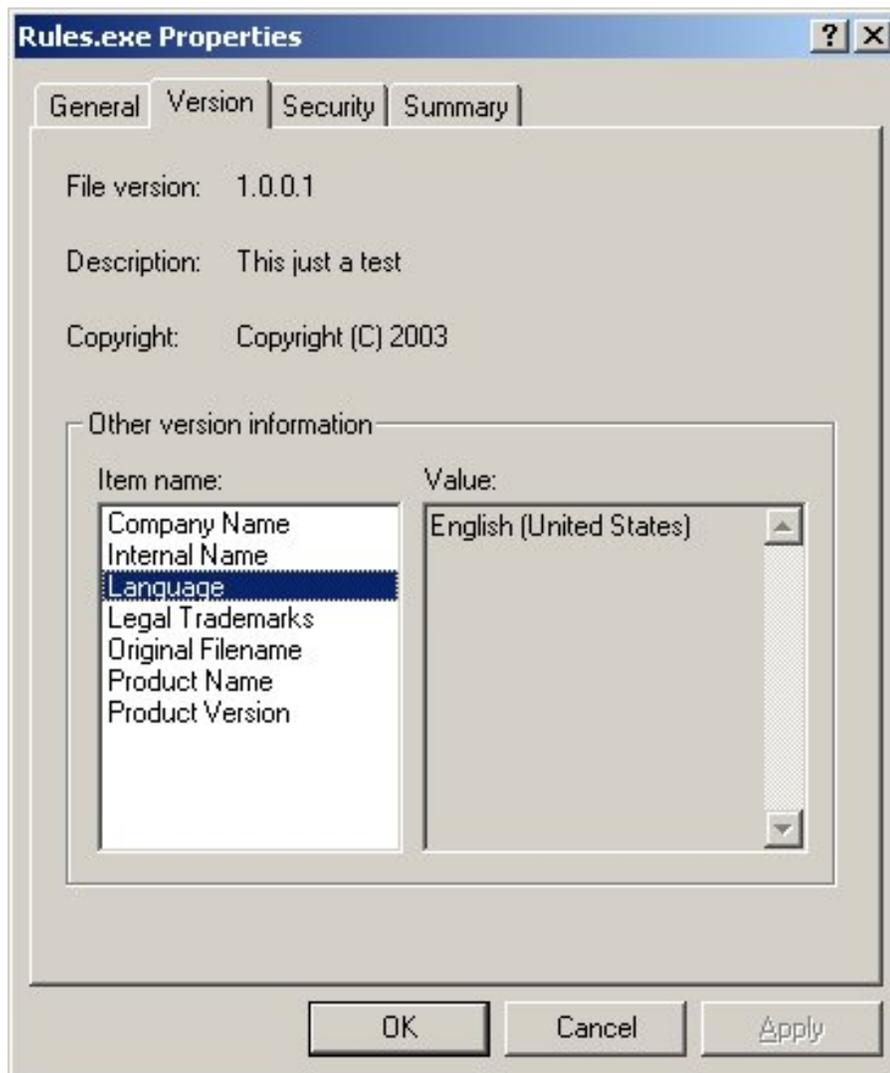
```

```

        VALUE "ProductName", "Version Info Script compiler\0"
        VALUE "ProductVersion", "1, 0, 0, 1\0"
    END
END
BLOCK "VarFileInfo"
BEGIN
    VALUE "Translation", 0x409, 1252
    VALUE "Translation", 0x419, 1251
    VALUE "Ã'Ã...Ã'Ã' ", 0x409, 1111
END
END
END

```

If you include version information in your DLL or EXE, you can right-click on that file in Windows Explorer and you'll see a versions tab. For example, if you save the above text into Rules.Version (although unlike manifests, the name part need not match the application name), include that file in the Rules application's project, and compile, right-clicking on Rules.exe will show the version information in Figure 10.



**Figure 10. Version information**

## Other improvements

Besides the IDE improvements, the source code and template changes, and the new examples, there are also a number of changes to the Clarion language. I've alluded to some of these, mainly in [Part 2](#). To see SoftVelocity's listing of C6 changes, go to their [C6 web page](#). By the way, there are a number of Flash videos in the C6 bin\Flash directory, although it's not immediately clear to me where those listed in the C6 help.

## Summary

This isn't a formal review, but rather an extended first look at Clarion 6. What I've seen so far, and have heard from other Clarion developers, indicates that this is a stable release with more than enough new features and bug fixes to justify the cost of an upgrade. Some core features, most notably the threading library, have been completely rewritten, and there are numerous additions such as ADO support, the graphing library, XML libraries, SQL blob handling and ODBC synchronization, improved browses, report output options, and more. Legacy developers have not been abandoned either, as ABC features are migrated into the Clarion template set (using ABC classes under the hood).

If you haven't already upgraded to C6, I suggest you give it serious consideration. Hang around the SoftVelocity [newsgroups](#) and see what other developers are saying. And note also what they aren't saying; although some inevitable post-release bugs have been found, this appears to be a very smooth upgrade for the majority of developers.

---

*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](#)

**Thank you. I read all 4 articles, they are very useful and...**

**Gee, Dave, remember Richard Taylor always recommends doing...**

online sales and delivery  
for your applications & tools

Developer **PLUS**

## Moving Applications to Oracle: RI And AutoNumbering Part 4

by Jon Waterhouse

Published 2003-12-11

[Last week](#), while discussing transactions, I stated that I have been a bit slack in my use of the word *save*. Using flat files you think about things the following way: you grab the record off the disk into the record buffer in memory; you make the changes in memory; you save the changed buffer back to the disk. Game over. Anyone else comes along afterwards and reads from the disk, they see the changed record.

With Oracle, that is not true. If I (User1) grab (select) a record for update and make a change using the UPDATE command (e.g. UPDATE Customer set firstname='Jon' where lastname='Waterhouse' and firstname='John') and you (User2) now come along and query the database to fill your browse `Select id, firstname, lastname from customer where lastname like 'W%'` you will still see the John Waterhouse row. You will continue to see the John Waterhouse row, and not the Jon Waterhouse row until I (User1) COMMIT my change (and maybe not even then - see below). I on the other hand, filling my browse with the same query *before* I commit will see the row as Jon Waterhouse and *not* John Waterhouse. I made the change; I get to see the change. But no one else gets to see the change until I decide I'm happy with it (and any other changes I might have made), and COMMIT. So, when I have used the word *save* in the previous discussion on locking, what I meant was update *and* commit.

With this extra bit of knowledge, let's revisit the discussion on locking. If my Clarion application does not use the FOR UPDATE clause to implement locking, and there is a large delay between UPDATE and COMMIT carried out by User2, then a simple re-query before our UPDATE accomplishes virtually nothing; I would only see the record had changed if my own session had changed it, or (possibly) if user 2 had got around to committing her work.

Twice now I have mentioned that even if another user commits, my session might not see those changes. Why is this? In some really complicated transactions (say the calculation of accounting balances based on all entries in an accounting system) you may try to read the same data twice. If the second time you read the data you get back different information than the first time this may make it impossible for your calculations to produce a result that adds up. This sort of transaction needs to be protected from reads that are "non-repeatable" (i.e. because someone else has changed the data meantime) or "phantom" (someone has added a record that didn't exist the first time we read the data). There is a third sort of read, called "dirty", which allows you to see uncommitted data, but Oracle does not allow this under any

circumstances. The Transaction Isolation Level determines what sorts of reads are allowed. The two choices in Oracle are Read Committed, which allows both non-repeatable and phantom reads, and Serializable, which allows neither.

That said, what does Clarion actually do? Does it issue FOR UPDATE selects? Is data committed immediately after any and all SQL statements that update data? What transaction isolation level is used?

In the standard browse-form application, when you select a record for update, the TakeEvent method of the BrowseManager first calls the Window.Update method, and then the Ask method calls the update form procedure. It is the Windows.Update method that makes sure that all of the buffers for the browses on the window are refreshed from the underlying views. This translates into a straight SELECT \* (all columns) statement. The process is summarized in the following table:

<b>In the browse, when you select Change on the browse&gt;</b>	
BrowseManager.TakeEvent	Chooses what to do based on which control has been accepted. In the case of the Change button, does the below:
Window.Update	Calls UpdateViewRecord for all the browses on the window, which GETs the current record
BrowseManager.Ask	Opens the form defined as the update procedure
<b>In the form, when you click the OK button (for an update)</b>	
Window.TakeCompleted()	Calls different routines depending on whether the form was called to insert, change or delete
RelationManager.Update	Initiates the logout (translates into setting the transaction isolation level to Serializable, and turns off auto-commit)
FileManager.TryUpdate/ UpdateServer	Tries to save the primary record using PUT. This translates into an UPDATE statement in SQL; passes errors, concurrency errors in particular, back to the caller
RelationManager.Update (continued)	If saving the main record succeeded, and RI is to be handled locally, calls the SecondaryUpdate method (which may be called recursively) to update the child records if the linking value has changed.
	The COMMIT command changes the transaction isolation level back to Read Committed, and turns auto-commit back on.

In the form procedure, when you press the OK button, a complicated sequence of Window methods is called. For purposes of this discussion, it's sufficient to note that if everything goes according to plan the `TakeCompleted` method calls a routine based on whether the form was called to update, insert or delete. In the case of an update the method checks to see if any changes have been made, and if so it calls the `SELF.Primary.Update()` method, i.e. the `RelationManager` method for the primary file. This method in turn uses the `UpdateServer` method of the underlying `FileManager`. At the beginning of the `RelationManager` method, a `LOGOUT` is performed (what happens here is explained below), and in the `FileManager` method an SQL `UPDATE` statement is prepared as a result of the `PUT()` statement of the form:

```
Update mytable set changedfields=:newvalues
  where changedfields=:oldvalues
  and allotherfields=originalvalues,
```

That is, Clarion implements an optimistic concurrency check. If this statement succeeds the `RelationManager` method goes on to make changes to child files (if the RI enforcement is local and not on the server). Finally, at the end of the `RelationManager.Update` method, assuming that no errors have been encountered at any stage, the Clarion `COMMIT` statement changes the isolation level back to `Read Committed` and turns auto-commit back on, which effectively commits the transaction.

If that description was a bit much, then here is just the bottom line. Clarion uses optimistic concurrency checking. The scope of a transaction is from the beginning to the end of the `RelationManager` method; in the middle of the process a call to the `FileManager` method actually writes the changes. Changes are committed at the end of the `RelationManager` method. The transaction isolation level is *serializable*.

One of the advantages of Clarion is that you can use the same syntax and have the database drivers convert your Clarion statements into statements appropriate for the particular file driver. And even with Oracle as the back end, you have (with the Enterprise edition) a choice of file drivers to talk to Oracle, the generic ODBC driver and the Oracle Accelerator.

ODBC, by default, sets auto-commit on. This means that every statement sent to Oracle is implicitly followed by a commit. With auto-commit on there is no way that you could implement a consistent multi-statement transaction. Thus, one of the most important things that the `LOGOUT` statement does is to change the ODBC Connect environment setting to turn auto-commit off. This is important to know. If, for example, you wanted a statement to be part of the same transaction as the main statement, and you placed it *before* the `RelationManager.Update` procedure, since auto-commit is, by default, turned on, this statement would auto-commit. If the main statement later rolled back because of errors, your initial, pre-logout change would still be stored in the database.

The second ODBC operation carried out by the `LOGOUT` is to set the transaction isolation level to `SERIALIZABLE`. By default, Oracle works with an isolation level of `READ COMMITTED`. In most cases the change in isolation level won't make any difference to you, as changes to the one record you are working with are dealt with by the concurrency check; it is only if your transaction includes reading other data that this will affect you.

Oracle aficionados will tell you that the reason ODBC defaults to auto-committing every update is that

it's a Microsoft design, and that Microsoft's SQLServer quickly falls to its knees when transactions remain uncommitted because it can't handle large numbers of unreleased locks. They also maintain that pessimistic concurrency checking (which entails locks being held for longer) is a more sensible approach than optimistic checks, but again that optimistic checking is chosen because of the locking problems of non-Oracle databases.

When you are using Oracle Accelerator the only real difference in the processing of transactions compared to the ODBC method is that the transaction isolation level stays at read committed.

Similar events take place for inserts and deletes.

The important thing to understand about the way that inserts, changes and deletes are accomplished in Clarion is that if you want to make other changes as part of the same transaction, it has to be between the LOGOUT and COMMIT statements, and this means your code has to happen in the middle of the existing ABC RelationManager (RM) methods. The obvious place to add code is after the FileManager (FM) method that is called by the RelationManager method. The relevant methods are the TryUpdate, TryInsert and DeleteRecord methods. Your additional code would first check to make sure the main action happened, then make additional changes.

It is quite possible that you want to add different things into the transaction depending on the context. For example, if the application maintains a transaction journal file, which is the main file that is updated by a few screens for varied purposes, the context somehow has to be communicated to the method. This might involve adding a parameter to the FileManager methods and the methods that call them.

I should note that the above approach is based on the C5.5 templates, which make the basic change (additions to all transactions on a file irrespective of context) relatively easy to accomplish. In the C5 templates things are not quite as easy. This is because the insert and delete methods call the FileManager UpdateServer and InsertServer methods, which are private, and the RM.Delete method issues its own DELETE command, while in C5.5 the new FM.DeleteRecord method is called.

## Getting more complicated

Back at the beginning of [Part 3](#) I brought up the classic transaction of the transfer of money between two bank accounts. I still haven't got beyond the simple transactions that Clarion manages itself, plus some alternatives in getting Oracle to manage some of the RI. Before going on to discuss multi-element transactions, I'd just like to mention one other Oracle feature that may help you avoid entering this realm. Clarion, in its documentation of views, says:

PUT only writes to the primary file in the VIEW because the VIEW structure performs both relational Project and Join operations at the same time. Therefore, it is possible to create a VIEW structure that, if all its component files were updated, would violate the Referential Integrity rules set for the database. The common solution to this problem in SQL-based database products is to write only to the Primary file. Therefore, Clarion has adopted this same industry standard solution.

However, if your view is declared in Oracle, you will find that you *can* directly update most of the

columns in the view, whether they are in the primary file or not. There are some exceptions. For example, if your view contains a "virtual column" calculated from one or more table columns (for example, the view contains a NAME column that is based on concatenating the FIRSTNAME and LASTNAME columns in a table), you cannot directly update it. With an Oracle view set up, you can use the view as the "file" for a form, and Oracle can update the view fields you change on the form regardless of the underlying table they come from. In some cases this can dispense with the need to set up multiple data update statements in a transaction.

Here's an example. You have three tables: MACHINES, PARTS\_LIST and PARTS. You could set up a view in Oracle like this:

```
CREATE VIEW v_machine_cost AS
SELECT m.machine_name, p.part_name,
       l.num_parts,p.unit_cost,
       p.unit_cost*l.num_parts extended_cost
FROM machines m,parts_list l,parts p
WHERE m.id=l.machine and l.part=p.id
and m.id=:machine_of_interest
```

A form based on this view would allow you to change how many of a particular part is required for the machine and the unit cost of the part. Oracle is smart enough to translate this into a change in the PARTS\_LIST table for the first and change in the PARTS table for the second. This gets rid of the need to write two update statements arising from the changes on one window.

## Summary

In this series of articles I've some of the issues involved in moving an application from flat ISAM files to a client-server architecture using a database like Oracle as a back end. As should be clear by now, it is not just a matter of setting up the Oracle tables and pointing your dictionary at them. Both relational integrity and autonumbering are essential aspects of migration.

---

*[Jon Waterhouse](#) has been using Clarion since the 2.1 days. His main work is as an economist, and he finds that Clarion is well-suited for applications which impose order on various sets of data. His projects include questionnaire data entry programs, classification software (assigning projects to groups), plus some more interesting scheduling applications. Jon has also used Clarion to link text information together, and is currently developing a program that will store linked snippets of WordPerfect documents and print custom documents composed of several of these snippets. He is currently working for the Newfoundland Government on a project to measure the performance of government employment programs.*

## 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

online sales and delivery  
for your applications & tools

Developer **PLUS**

## Save and Restore Window Size and Location

by Nardus Swanevelder

Published 2003-12-31

One of the most recent requests I received from users was to be able to resize the application's windows, and the application had to remember the resizing as well as the window's position on the screen. Furthermore, the users wanted to be able to reset to the last-saved settings.

I am using [Easy Resize and Split](#) from IngasoftPlus, and I thought this should be a breeze. But Easy Resize and Split only remembers the sizing of the screen and not the position of the screen, since that's something the standard templates already handle. I am also using the [Icetips Cowboy SQL Templates](#) which give the user the option to change the column sizes in a browse box, but it does not have a reset function by default. What I really needed was an integrated solution.

After some time reading and searching for a solution I found the following information, in regards to the Global Properties of an application, in the Clarion help:

### **Use .INI file to save and restore program settings**

Check this box to have your application create and maintain a standard .INI file. Many of the Clarion templates are designed to take advantage of the .INI file, for example, to save and restore window size and location.

### **.INI File to use**

Specify whether to use the default.INI file name. By default, the application creates the .INI file with the same file name as your application. To use another name select the Other choice from the drop down list.

### **Other File Name**

Specify the other .INI file name.

This looked like something that could be the solution to my problems. But what changes are made to an ABC application when the above settings are activated?

The following code is added to the `ThisWindow.Init` embed:

```
INIMgr.Fetch( 'Update_AdhocProduct' ,QuickWindow)
```

`Update_AdHocProduct` is the procedure name and `QuickWindow` is the label of the window used in this procedure.

What does the Clarion help say about this command:

**Fetch(section,window)**Retrieves and restores several WINDOW attributes saved by a prior corresponding call to Update(section,window) from the INI file specified in the INIClass.FileName property. Restoring the values returns the specified WINDOW to its saved position and size.

According to the Clarion help the application should have an update command as well. This update command can be found in the `ThisWindow.Kill` embed and it looks like this:

```
IF SELF.Opened  
    INIMgr.Update( 'Update_AdhocProduct' ,QuickWindow)  
END
```

The `Fetch` and `Update` methods retrieves and saves the following values for the window: `Maximize`, `XPos`, `YPos`, `Height` and `Width` from the system ini file or from the other INI file specified under the `Global` properties.

That takes care of ABC but what is saved by the Easy Resize and Split template?

Easy Resize and Split saves firstly the width and height of the window, and when vertical and or horizontal bars are added to a screen the template also saves the Original Position and Length as well as the current position.

Following is a sample INI file - `Update_Quote` is the current procedure name and the following lines contain the information that the ABC templates will save about the Window:

```
[Update_Quote]  
Maximize=No  
XPos=73  
YPos=48  
Height=291  
Width=531
```

The lines above save the window position and size used by the Clarion templates to remember the position of the screen.

The next two lines contain the settings for the basic Easy Resize and Split template.

```
EasyResize:OrigWidth=805
```

```
EasyResize:OrigHeight=507
```

The next three lines store the settings that the Easy Resize and Split template requires to keep track of the vertical split bar on the window.

```
EasyVSplit:EasySplit:OrigPos=210
EasyVSplit:EasySplit:OrigLen=5
EasyVSplit:EasySplit:Pos=210
```

In the example above the use variable for the Vertical split control is ?EasyVSplit and the rest of the INI entry value is made up of one of the following: EasySplit:OrigPos or EasySplit:OrigLen and EasySplit:Pos. Remember that the Easy Resize and Split template has a horizontal split bar as well.

As I mentioned earlier I am also using the IceTips Cowboy SQL template, which allows you to resize the columns on a browse and save that info from session to session.

To understand how the SQL template implements this function, search for the following topic in the Clarion help: *How to Display the Sort Field First on a Multi-Key Browse*. This will explain how to save the browse column format and it also explains how to alter it.

The following is what the SQL template can save to an INI file, for example:

```
SQL18FORMAT=32C(2)|M*~Item~@n3@62L(2)|M*~Product Code~S(200)␣
@s200@95L(2)|M*~Product Description~S(900)@s255@18C(1)|␣
M*~Qty~C(2)@N-5@45D(1)|M*~Total Cost~C(2)@n-12@22C(7)|␣
M*~GP~C(2)@n-6.2@55D(10)|M*~Total Sell~C(2)@n-15.2@37D␣
(24)|M*~Exchange~C(2)@n9.5@40D(1)|M*~AEL Fee~C(2)@n-11@40D␣
(1)|M*~Installation~C(2)@n-11@40D(1)|M*~Maint.~C(2)@n-11@Q␣
'Maintenance Amount'40D(1)|M*~Discount~C(2)@n-11@
SQL18COLUMN=1
SQL14FORMAT=
SQL14COLUMN=
```

The lines above tell you that there are two list boxes on this window, one is referenced by SQL18 and the other by SQL14. The format entry saves the list box format in terms of column sequence and size and the Column entry saves the default search column.

Being a member of the Lazy Programmer's Society, I knew I didn't want to have to keep writing code to handle each different situation. The problem demanded the creation of a template.

First, take a look at what the template should do:

1. Alert a key which the user should press to reset the settings
2. Display a message to the user asking if he/she wants to reset the settings
3. Reset the settings
4. Display a message after successfully resetting the settings

## The template

The template should be a procedure extension template so that I can add it to my procedures.

For each procedure it should have a window caption for the message box and it should know what key will be used to activate the screen reset.

```
#EXTENSION(QualitasResetScreenFormat,'Qualitas Reset Screen Format')
  ,PROCEDURE
#SHEET
#TAB('Reset Screen')
#DISPLAY (')
#DISPLAY (')
#DISPLAY ('What is the Caption of the Message Box?')
#DISPLAY ('(E.g. "Client Update"')
#PROMPT ('',@s255),%MessageCaption,REQ,DEFAULT(%Procedure)
#DISPLAY (')
#DISPLAY (')
#DISPLAY ('Which key do you want to Alert on the Window?')
#DISPLAY ('(E.g. "CtrlL"')
#PROMPT ('',@s255),%AlertKeyonWindow,REQ,DEFAULT('CtrlL')
#DISPLAY (')
#DISPLAY (')
```

For each list or browse on the screen the following fields are presented:

- What is the use variable for the list or browse. This is a required field.
- What is the Description for the list or browse. This is a required field.
- What is the Message Box Caption for the list or browse. This is not required as the template will use the Description of the list or browse if the caption is empty

```
#BUTTON ('Add descriptions for SQL List Boxes'),
  MULTI(%SQLDescription, %ListUseVariable & ' - '
    & %OwnDescription & ' - ' & %SQLMessageCaption), AT(10,, 180)
#PROMPT ('Use Variable',CONTROL),%ListUseVariable,REQ
#PROMPT ('Description',@s255),%OwnDescription,REQ
#PROMPT ('Message Box Caption',@s255),%SQLMessageCaption
#ENDBUTTON
#DISPLAY (')
#DISPLAY (')
#DISPLAY ('Includes Support for IceTips SQL templates')
#DISPLAY ('Includes Support for IngaSoft Easy Resize & Split')
#ENDTAB
#ENDSHEET
```

The following code defines all the variables that the template uses. LCL:INI\_Reset is used to determine if the screen settings should be saved and the other two variables are used to determine if a message should be displayed stating that the settings were reset.

```
#AT(%DataSection),PRIORITY(4000)
LCL:INI_Reset          BYTE(0)    ␣
    !Flag to determine when screen settings saved
LCL:INI_ScreenReset    BYTE(0)    ␣
    !Flag determine screen reset mess display
LCL:INI_BrowseReset    BYTE(0)    ␣
    !Flag determine browse reset mess display
#ENDAT
```

The code below alerts the key that will be used to activate the reset code. The alert is set on the window so that it is possible to execute this code from anywhere on the screen.

```
#AT(%WindowManagerMethodCodeSection,'Init','()',BYTE'),␣
    PRIORITY(9050)
    %Window{Prop:Alrt,255} = %AlertKeyonWindow
#ENDAT
```

The next code is responsible for displaying a message box asking the user if the screen or SQL List Boxes settings should be reset.

```
#AT(%WindowEventHandling,'AlertKey'),PRIORITY(4500)
Declare some local variables for the template
#DECLARE(%IceTips1)
#DECLARE(%IceTips2)
#DECLARE(%ListControl)
#DECLARE(%ListDescription)
#DECLARE(%SQLMessageBoxCaption)
#DECLARE(%EasySplit)
```

Check if the programmer has specified an alert key.

```
If KeyCode() = %AlertKeyonWindow
    #EMBED(%BeforeMessageResetStart,␣
    'Before Message that layout is going to be reset')
    #!Ask if screen layout must be reset
```

If an alert key has been pressed, display a message box and ask the user if he/she wants to reset the settings for the screen. The message box will use the caption specified by the template prompt.

```
LCL:Ini_ScreenReset = 0
CASE MESSAGE('Are you sure you want to reset the ␣
    layout of the screen?','%MessageCaption',ICON:Question,␣
    BUTTON:Yes+BUTTON:No,BUTTON:No,1)
OF BUTTON:No
OF BUTTON:Yes
    #EMBED(%BeforeCheckingINIFileName,␣
    'Before Checking INI File Name')
```

If the user wants to reset the screen settings, call the `Check_IniMgrFileName` procedure that check if the INI manager's file name is empty. If the filename is empty get the system default INI file name and initialise the INI Manager with it:

```
#EMBED(%BeforeCheckingINIFileName, ←
  'Before Checking INI File Name')
Do Check_IniMgrFileName
#EMBED(%AfterCheckingINIFileName, ←
  'After Checking INI File Name')
LCL:Ini_ScreenReset = 1
```

Set the variable that tracks if the ABC code that updates the screen settings should be executed.

```
LCL:Ini_Reset = 1
```

If the Easy Resize and Split template has been used on this procedure, update the Easy Resize and Split settings in the INI file with blank values:

```
#!Reset EasyResize
#FOR(%ActiveTemplate), ←
  Where(%ActiveTemplateType = 'EXTENSION')
  #IF(%ActiveTemplate='EasyResize(EasyResizeAndSplit)')
INIMgr.Update('%Procedure', 'EasyResize:OrigWidth', '')
INIMgr.Update('%Procedure', 'EasyResize:OrigHeight', '')
  #FOR(%ActiveTemplate), WHERE(%ActiveTemplate= ←
    'EasyHSplit(EasyResizeAndSplit)' OR ←
    %ActiveTemplate= 'EasyVSplit(EasyResizeAndSplit)')
  #FOR(%ActiveTemplateInstance)
    #FIND(%ControlInstance, %ActiveTemplateInstance, ←
      %Control)
```

Remove the "?" from the use variable name

```
  #SET(%EasySplit, SUB(%Control, 2, LEN(%Control)-1))
INIMgr.Update('%Procedure', '%EasySplit' & ':EasySplit:OrigPos', '')
INIMgr.Update('%Procedure', '%EasySplit' & ':EasySplit:OrigLen', '')
INIMgr.Update('%Procedure', '%EasySplit' & ':EasySplit:Pos', '')
  #ENDFOR
#ENDFOR
#EndIF
#ENDFOR
```

Update the screen settings in the INI file with blank values:

```
INIMgr.Update('%Procedure', 'Maximize', '')
INIMgr.Update('%Procedure', 'XPos', '')
INIMgr.Update('%Procedure', 'YPos', '')
```

```

    INIMgr.Update('%Procedure','Height','')
    INIMgr.Update('%Procedure','Width','')
END
#EMBED(%BeforeResetSQLStart,'Before the SQL browse

```

If the SQL template has been used on this procedure check the template multi value variable to see if any list or browse should be reset:

```

#! Reset SQL Browse
LCL:Ini_BrowseReset = 0
#EMBED(%BeforeCheckingINIFileName1,␣
    'Before Checking INI File Name1')
#! Check again because user could ␣
    have said No to reset screen
Do Check_IniMgrFileName
#EMBED(%AfterCheckingINIFileName1,␣
    'After Checking INI File Name1')

```

Check if the active template is of type Control:

```
#FOR(%ActiveTemplate), Where(%ActiveTemplateType = 'CONTROL')
```

If the active template is of type Control, check if the active template is SQLBrowseNL (CCSAbc):

```

#IF(%ActiveTemplate='SQLBrowseNL(CCSAbc)')
#FOR(%ActiveTemplateInstance)
#FOR (%Control),WHERE(%ControlInstance = ␣
    %ActiveTemplateInstance)
#CASE (%ControlOriginal)
#OF ('?List')

```

If the control is a list, save the list's use variable by using the template variable %Control.

```
#SET (%ListControl, %Control)
```

Check the use variable against the multi value template variable and if it is the same and the description is not empty update the SQL template settings in the INI file with blanks.

```
#FOR(%SQLDescription),WHERE(%ListControl = %ListUseVariable)
```

If the programmer gave the browse or list box a description:

```

#SET (%ListDescription, %OwnDescription)
#IF(%ListDescription <> '')
#IF(%SQLMessageCaption = '')

```

If the programmer did not specify a message box caption, populate the default caption.

```

        #SET (%SQLMessageBoxCaption,%ListDescription & ' Browse')
#ELSE
        #SET (%SQLMessageBoxCaption,%SQLMessageCaption)
#ENDIF
CASE MESSAGE('Are you sure you want to reset
the layout for the %ListDescription Browse?',
'%SQLMessageBoxCaption', ICON:Question,
BUTTON:Yes + BUTTON:No, BUTTON:No, 1)
OF BUTTON:No
OF BUTTON:Yes

```

The code that is used below makes use of a feature that is not available in the current released version of the SQL templates. This feature allows you to save the list or browse settings, per user, per list or browse, and not just per list or browse. This feature should be available in the next release from IceTips but if you are interested in this feature in the mean time please contact me and I will make the code changes available to you.

```

        #Set(%IceTips1,'SQL' & Clip(%ActiveTemplateInstance) & 'Format')
        #Set(%IceTips2,'SQL' & Clip(%ActiveTemplateInstance) & 'Column')
INIMgr.Update(Clip(GLB:User) & '%Procedure','%Icetips1','')
INIMgr.Update(Clip(GLB:User) & '%Procedure','%Icetips2','')

```

If you do not want to make use of this added feature remove the comment indicator in front of the code below to update the INI settings with blanks. Remember to add a comment indicator to the code above.

```

#! Use code below if SQL Template not supporting saving format per user
#! INIMgr.Update('%Procedure','%Icetips1','')
#! INIMgr.Update('%Procedure','%Icetips2','')

```

If the following code is not added to the template the Cowboy SQL Template will save the list or browse settings on closing the screen, thereby overriding your blanks with the current settings.

```

SQL%ActiveTemplateInstance.SaveBrowseFormat(False)
LCL:Ini_BrowseReset = 1
END
        #ENDIF
        #ENDFOR
        #END
        #ENDFOR
        #ENDFOR
        #ENDIF
#ENDFOR
#EMBED(%AfterResetSQLStart,
'After the SQL browse layout has been reset')
#EMBED(%BeforeMessageResetFinished,
'Before Message that layout has been reset')

```

If the user chose to change the settings display a message confirming that the settings were reset:

```

    If LCL:Ini_ScreenReset = 1 or LCL:Ini_BrowseReset = 1
        Message('The Layout was reset. Please close the screen ←
                and reopen to view the default layout', '%MessageCaption')
    END
    #EMBED(%AfterMessageResetFinished, ←
        'After Message that layout has been reset')
    END
#ENDAT

```

The following code makes sure that the ABC code for updating the screen settings is only executed if the user did not request a reset:

```

#AT(%WindowManagerMethodCodeSection, 'KILL', '()', BYTE') ←
    , PRIORITY(7200)
    If LCL:INI_Reset = 0
#ENDAT

```

```

#AT(%WindowManagerMethodCodeSection, 'KILL', '()', BYTE') ←
    , PRIORITY(7800)
    .
#ENDAT

```

Here is the code for the Check\_IniMgrFileName procedure.

```

#AT (%ProcedureRoutines)
Check_IniMgrFileName Routine
    If IniMgr.FileName = ''
        BRA:Branch_Code = GLB:Default_branch
        Access:Branch.Fetch(BRA:Branch_Key)
        If BRA:System_INIFileName = ''
            BRA:System_INIFileName = 'SalesSet.ini'
        END
        IniMgr.Init(Clip(Left(BRA:System_INIFileName)))
    END
#ENDAT

```

## Summary

In this article I've described how to create a template to add resizing/positioning reset capabilities to a procedure that uses the [EasyResizeAndSplit Templates](#) from Ingasoftplus, and the [Cowboy SQL Templates](#) from IceTips. With a little work you should be able to adapt this template to other products as well.

[Download the source](#)

[Nardus Swanevelder](#) was born and raised in South Africa. He was a networking engineer for seven years before he moved over to the commercial side of the business. Nardus has developed a [Sale Cycle Management system](#) for the Information and Communication Technology industry. Nardus has been programming in Clarion since 1989, and holds B.Com and MBA degrees.

## 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.



## Creating Utilities For MS SQL 2000

by Bernard Groperrin

Published 2003-12-31

Quite often it happens that Clarion programmers see only limitations when using SQL from within Clarion, as compared to other tools. In fact, this limit is mostly in our heads. Clarion's `PROP:SQL` makes it easy to execute arbitrary SQL statements in Clarion, and a few tricks simplify the task of getting information back from these SQL statements.

In this article I will show how you can create utilities for an SQL database with `PROP:SQL`, using the example of MS SQL 2000. If you are familiar with the Clarion community's most popular Internet sites, you may have noticed an excellent set of articles by Dan Pressnell on SQL, published on [Icetips](#). I will build on Dan's ideas to create a little utility for creating a Microsoft SQL 2000 database, including several tables, entirely from scratch.

**NOTE:** I have not tested the code with SQL server 7.0, but I can see no reason why it shouldn't work there, or with MSDE. In fact with some modifications you should be able to use this approach with just about any SQL database, as long as there are system tables.

### How it works

The basic idea behind this utility is that all that Clarion needs to send commands/requests to the server is a "dumb" file in the dictionary with a number of fields large enough to contain the result of the biggest request you think you may have to execute. There is absolutely no need for this file to match anything actually existing in the database, except for the name of an existing table. As long as you have this file declared, (some developers called it this the "stupid temp table trick") you can execute SQL statements on the server and get data back.

In order to achieve this goal as simply as possible, I will use two tools extensively. One, as I mentioned before, is Clarion's `Prop:SQL`, which allows me to send commands directly to the server; the other is what I would name the "Queue from Queue", or maybe more precisely the "Query from Queue" mechanism, which is a great idea from Dan Pressnell.

I will explain what this tool does in detail later on, but basically the Query from Queue mechanism does this:

1. It frees me from having to manually type (sometimes very lengthy) Select statements.
2. It returns the results of a query directly into a Clarion Queue, so that I can check this result set more than once, sort it different ways, check directly for a given value with a `get`, in other words all things not possible with the classic Clarion result set, which I can only read sequentially, and will lose on my next query.
3. It lets me use a column name in my code, as if I was using `Prop:SQL` on my dumb file. Ordinarily I would have to write something like `name=dumbfile.f1` somewhere, then later `phone=dumbfile.f1`, which does not help much making my code readable. With the Query from Queue technique, I can write `name = QueryQueue.name`, which makes a little more sense.

But first, I need to back up just a little. I said I am going to write a utility to initialize a database from scratch. But how do I connect to a nonexistent database?

Fortunately, even without any operational database on the MS SQL server, a model/template for all other databases already

exists, containing a certain number of system tables, plus many other things needed for the database engine to work properly, such as views, stored procedures, etc. This MS SQL database is named `master`.

So I want to connect to an MS SQL 2000 server, to the `master` database. I obviously need to know a user name and password with an administrator level to be able to create a database, whether I'm using Clarion or not. This being done, I want to create a database (without, if possible, overwriting an existing one with the same name), create a user who will be the owner of this base, create the tables I need, and check that I have done what I believed I was doing.

I must also say that I want this to be a simple process. I do not want to have to distribute an SQL script to my users which they need to run through some other utility; my users may have only MS Client installed on their computers, and nothing else.

## The dictionary

To simplify my life a little, in this first version, I created a dictionary by importing a few tables from the master database (although as I will show in a future article it's possible to do without these tables). Figure 1 shows the dictionary.

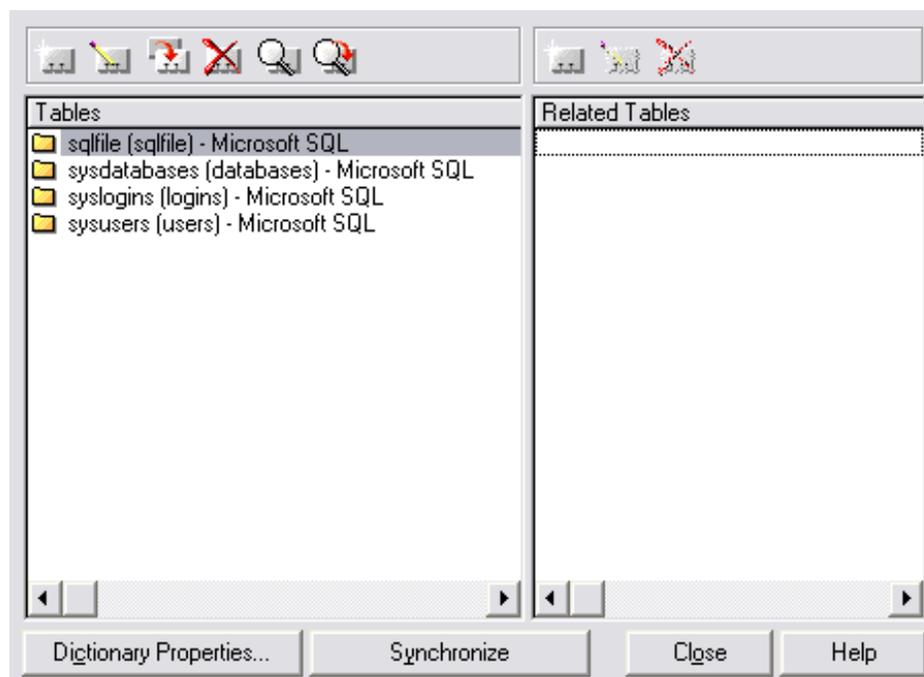


Figure 1. The necessary tables.

You can see that there are not so many tables needed:

Sysdatabases  
 Syslogins  
 Sysusers

`Sysdatabases` contains the list of databases, `Syslogins` the list of roles (a role is a set of access rights grouped under a name), and `Sysusers` is the list of users, who can be associated with multiple roles.

The more interesting table is the one called `SqlFile`. Figure 2 shows the layout.

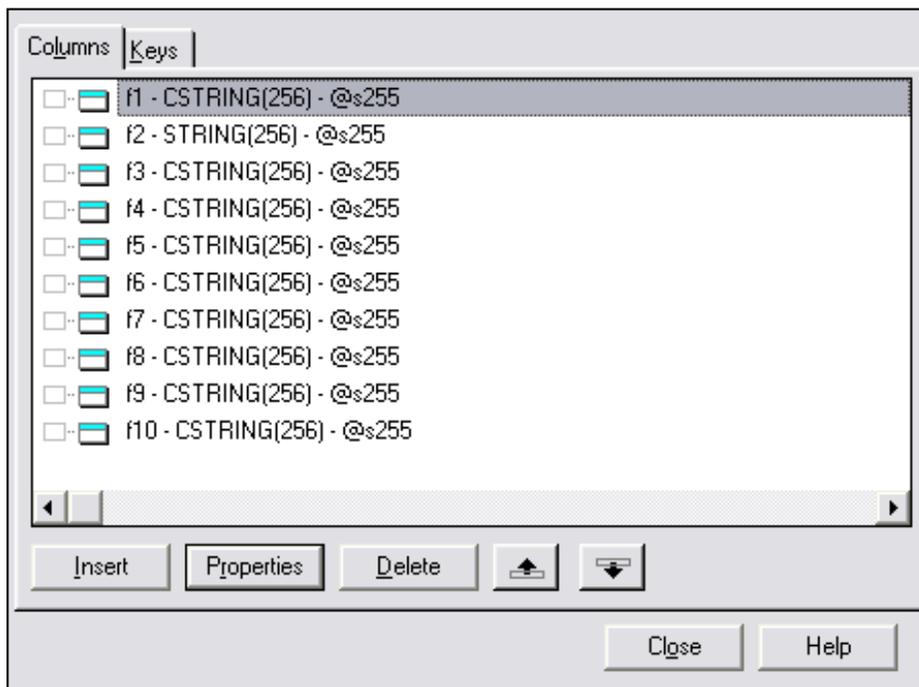


Figure 2. The SqlFile structure

SQLFile may look a little weird to you, even more so because the full path name for this table is sysdatabases, again! Clarion's "magic" data type conversion will allow me to use this table in a generic way, as a recipient for a *result set* (i.e. a set of records, using the specified columns) of an SQL request. There are only ten fields here, meaning any SQL statement cannot return more than ten columns of data, but nothing restrains you from creating more fields to accept result sets with as many fields as you like.

As this utility targets end-users, I have no idea what their server name could be, as well as their user name or password (if they had the good idea to change the default user name of sa to something a little less known and a bit more sophisticated), so there is no owner name specified anywhere in the dictionary. As a result, the application automatically opens the connection dialog box while you try to open any of the SQL tables.

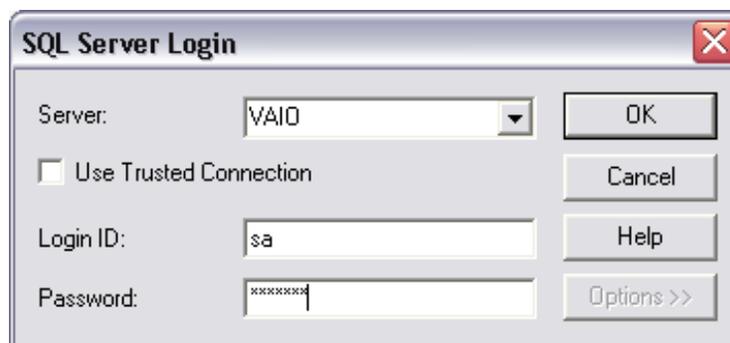


Figure 3. The login dialog box

And that is a ready-made dialog I don't have to do anything about – it's all handled by the driver.

## Send\_Query

One of the requirements of this application is the ability to send an SQL command to the server. So that I don't have to write the same code over and over again, here's a function that sends the query and reports any error:

```
Send_Query          PROCEDURE (string pQuery)
! Start of "Data Section"
```

```

! [Priority 5000]
! End of "Data Section"
CODE
! Start of "Processed Code"
! [Priority 5000]
open(sqlfile)
sqlfile{prop:sql} = pQuery
if errorCODE()
    Message('Error:' & ErrorCODE() & '-' & error() |
        & '<13,10,13,10>' & fileerror() & '<13,10,13,10>' & pQuery)
    Return 1
end
return 0
! End of "Processed Code"

```

I simply pass as parameters the SQL request I want to execute, do a `SqlFile{prop:Sql}`, and test if I get an error. As a result, I now have a single line in the code to test for error, instead of six, and I am going to execute quite a few requests.

### Query\_from\_Queue

Okay, it's time for a bit more serious stuff. Here, my goal is to have a request *generated* from a queue, and to have its result returned into the same queue. No no, don't start running away, it's not black magic and sorcery, even if it *is* very cool. All the credit goes to Dan Pressnell; I am just using the code and trying to explain. Here's the code for the `Query_From_Queue` procedure:

```

Query_From_Queue      PROCEDURE  (*Queue pQueue, string pExtra, |
    byte pDebug=0, long pLimit=0) ! Declare Procedure
! Start of "Data Section"
! [Priority 1000]
i      long
r1     any
r2     any
dstr   &idynstr
! End of "Data Section"
CODE
! Start of "Processed Code"
! [Priority 5000]
dstr &= newdynstr()
dstr.cat('select ')
if pLimit <> 0
    dstr.cat('top ' & pLimit & ' ')
end
loop i=1 to 100000
    r1 &= what(pqueue,i)
    if r1 &= null
        break
    end
    if i > 1
        dstr.cat(', ')
    end
    dstr.cat(who(pqueue, i))
end
dstr.cat(' ' & pextra)
X#=send_query(dstr.str())
IF pDebug
    Message('QUERY=' & dstr.str())

```

```

END
dstr.release()
free(pqueue)
loop
  next(sqlfile)
  if error() then break.
  loop i=1 to 100000
    r1 &= what(pqueue, i)
    r2 &= what(sqlfile.record, i)
    if r2 &= null
      IF pDebug
        stop('Not enough fields in sqlfile to hold intended number of columns!')
        halt
      ELSE
        break
      End
    end
    if r1 &= null or r2 &= null
      break
    end
    r1 = r2
  end
  add(pqueue)
end

```

Query\_from\_Queue receives a Queue as parameter, pQueue, a String, pextra, a flag to debug (which is an omittable parameter), and a limit indicator, omittable also. Here's the declaration for the queue:

```

DB_Q      Queue
DbID      like(databases:dbid), name('sysdatabases.dbid')
FileName  Like(databases:filename), name('sysdatabases.filename')
DB_NAME   Like(databases:name), name('DB_NAME(sysdatabases.dbid) AS DB_NAME')
End

```

I'll explain the name('DB\_NAME(sysdatabases.dbid) AS DB\_NAME') syntax in a bit.

In MS SQL, the sysdatabases table is structured as follows:

```

CREATE TABLE [dbo].[sysdatabases] (
    [name] [sysname] NOT NULL ,
    [dbid] [smallint] NOT NULL ,
    [sid] [varbinary] (85) NULL ,
    [mode] [smallint] NOT NULL ,
    [status] [int] NOT NULL ,
    [status2] [int] NOT NULL ,
    [crdate] [datetime] NOT NULL ,
    [reserved] [datetime] NOT NULL ,
    [category] [int] NOT NULL ,
    [cmptlevel] [tinyint] NOT NULL ,
    [filename] [nvarchar] (260) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
    [version] AS (convert(smallint, databaseproperty([name], 'version')))
) ON [PRIMARY]

```

And here is the SELECT statement I want Query\_from\_Queue to build:

```
SELECT sysdatabases.dbid, sysdatabases.filename,
       DB_NAME(sysdatabases.dbid) AS DB_NAME
```

As you can see, the `SELECT` statement fields are the same as the fields declared in `DB_Q`. Let's see how `Query_From_Queue` builds the `SELECT` statement directly from this queue structure.

First, the code begins by initializing a dynamic string (the `Dynstr` interface) which will hold the `SELECT` statement. This is a string you can add to incrementally using its `cat()` (for concatenate) method. The first string added is `'select'`. If a limit on the number of records to return is specified, the next string added is `'top ' & pLimit & ' '`. Then comes the loop to construct the list of fields to return, using `WHAT`:

```
R1 &= what(pqueue, i)
```

The `WHAT` function returns a reference to the specified field number of a group, a queue or a record. Note `r1` is declared as `ANY`, so that it can receive a reference to any data simple type. To say it differently, at the first loop iteration within the queue, `R1` will receive the value `DB_Q.DbID`. If "what" returns nothing (null), the loop immediately breaks, as I reached the queue end. Using `WHAT`, here, has no other goal than letting the loop know when to break.

Note this code: `If i > 1 ; dstr.cat(', ') ;End`. In other words, the comma is only added after the second iteration, since `SELECT ,field1,field2` would result in a syntax error..

Next is the code `dstr.cat(who(pqueue, i))`. `WHO` returns a group's field name, from the position indicated by the second parameter. In the first iteration, `i` value is 1, and field's name is `sysdatabases.dbid`, so now the string contains the value `SELECT sysdatabases.dbid`.

In the second iteration `i > 1`, so the code adds a comma and second field's name; the value is now `SELECT sysdatabases.dbid, sysdatabases.filename`, and so on.

The loop exits after the third iteration (for this table), and `pextra` is appended. From the `Load_DBQ` routine in the main procedure you can see that `pextra` contains `FROM dbo.sysdatabases (result of Name(sysdatabase)) ORDER BY dbid`. Now the string's value is `SELECT sysdatabases.dbid, sysdatabases.filename, DB_NAME(sysdatabases.dbid) AS DB_NAME FROM dbo.sysdatabases ORDER BY dbid`.

Run this request in SQL Query Analyzer; you'll see something like the following:

1	C:\Program Files\Microsoft SQL Server\MSSQL\data\master.mdf	master
2	C:\Program Files\Microsoft SQL Server\MSSQL\data\tempdb.mdf	tempdb
3	C:\Program Files\Microsoft SQL Server\MSSQL\data\model.mdf	model
4	C:\Program Files\Microsoft SQL Server\MSSQL\data\msdbdata.mdf	msdb
5	C:\Program Files\Microsoft SQL Server\MSSQL\data\pubs.mdf	pubs
6	C:\Program Files\Microsoft SQL Server\MSSQL\data\northwnd.mdf	Northwind

By the way, `DB_NAME` is an MS SQL function which returns the database name from the database ID passed as parameter. In this case I really don't need to do that, as a field in `sysdatabases` contain this data, but it makes the data more readable.

Next, the code reads the "dumb" `SqlFile` table in a loop, and it starts again in a new loop very similar to the first one. Only this time, it copies the label from the `Queue` to `r1`, and the label from `sqlfile` to `r2`. It then tests for an error to exit the loop, and if there is no error it ends up with `r1 = r2`, or `DB_Q.DbID = Sqlfile:f1`, for the first queue field.

After the break, the code does an `add(pqueue)`. Et voilà !. This code is actually filling in the queue with the results matching the request generated by this very queue.

This explanation was a bit lengthy, but it's not very often that you see a queue used to generate an SQL request *and* containing the result of the request.

Now I have the tools for the job, and a pretty good idea on how to use them, it's time to do some real work.

## Main procedure

To be certain that the database I am going to create does not exist already on the server, the first thing to do is to load a queue with all existing databases on this particular server. MS SQL has a table, `Sysdatabases`, with this information. `Load_DbQ` routine is doing that job.

Next, the code finds the default path to create the database "physical files". In order to achieve this, it will simply look where the master database is installed, and filter the result to keep only the directory name.

The code then checks that the current user has the proper rights required to create a database on the server. The simple SQL `SELECT IS_SRVROLEMEMBER('sysadmin') AS Value` does all the work! `IS_SRVROLEMEMBER` is one of the very numerous stored procedures already in MS SQL (to get ideas about how to use these stored procedures, read the online books). The result of this request is 1 or 0, `true` or `false`, and is stored in a local variable. If the current user does not have the required rights, she is politely told so when the window opens.

The next test is to verify that the database does not exist already. This is very simple to implement, as the queue of the existing databases is already there. A simple `GET` on the queue determines if the database exists. If no error, there's a problem!

Now it's time to create the new database, using an SQL command built up from variables. In this example there are a certain number of parameters hard coded, but everything could be a variable as well. Read Microsoft documentation on the possible variants for `CREATE DATABASE`. Just remember that in my specifications, I wanted something simple for the end user.

It's always good to check that the database has actually been created. The code again loads the database list with, but this time, there should *not* be an error on the `GET`!

Now it's time to connect to the newly created database (`USE database`). The code creates a special user who owns the tables, and gives that user a certain number of privileges. Here again, the best thing is to dig into Microsoft documentation, as your needs might be very different than the one taken into account for this example. Since this user will have `CREATE` privileges, the code can now create every table, eventually with indexes and keys. As in this example, it might be a good idea to use an ASCII File to log what happens, and display this log to the user.

## Summary

With Clarion's `PROP:SQL` it's easy to execute SQL statements on MS SQL Server, or just about any other SQL server. And Dan Pressnell's `Query_from_Queue` procedure is a great tool for building SQL statements and retrieving the results of those statements into a queue.

This paper's goal is really not to explain Microsoft's own version of SQL syntax, but simply to show that it is totally possible to use all these commands from Clarion. And remember that with `PROP:SQL` you are not limited to existing tables, it's even possible to use dynamics result sets (not existing statically in tables, but dynamically calculated at request's time). I will demonstrate that in my next article, which shows how to create a utility to restore MS SQL backups.

[Download the source](#)

## 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

**Reborn Free**

**CLARION**  
*online*

## Russ's UK Trip

Published 2003-12-31

The following photos of Russ Eggen's UK trip are courtesy of Mark Sarson.



Russ Eggen, signing express parcels for his fans



Matt Connew, John Davidson, and Russ Eggen



Russ Eggen and Mark Sarson



Kathleen Eggen and Joanne Sarson

## Reader Comments

[Add a comment](#)

**Where am I then**

**Ian, In the phone pictures perhaps? They were a bit too...**

**The phone pictures were taken by me anyway so there...**