

INVEST

in your own abilities

Clarion
magazine

published by
CoveComm Inc.

Clarion MAGAZINE

[Search](#)

[Home](#)

[COL Archives](#)

[Information](#)

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)

[Finding A File Using Template Code](#)

The Clarion template language is more powerful than most developers realize. In this article Kevin Erskine shows how he wrote template code to locate and parse the Clarion redirection file so he could parse his own class includes at generation time.

Posted Friday, August 31, 2001

[In The Red](#)

In accounting, it's standard practice to show negative numbers in red. Here's a source procedure that will search the currently active window for any numeric fields capable of displaying negative numbers.

Posted Wednesday, August 29, 2001

[Migrating The Inventory Application To SQL Server \(Part 3\)](#)

In this three part series, Ayo Ogundahunsi takes the Clarion example Inventory application from a flat file TPS database to a MS SQL database, focusing on portability, business rules, relational integrity, and Clarion as an interface tool.

Posted Tuesday, August 28, 2001

[Weekly PDF for August 19-25, 2001](#)

All Clarion Magazine articles for August 19-25, 2001 in PDF format.

Posted Monday, August 27, 2001

[Subscribe to Clarion Magazine](#)

You can save 25% on back issue purchases during our Summer Back Issue Sale. Just follow the link to the order form (login required) and if you're missing any back issues, the order form will calculate the cost and apply a 25% discount. If you don't see any back issue options on the order form, that means you're all caught up. You can also use this form to extend your ClarionMag subscription.

[SealSoft xQuickFilter v2.0.4 Released](#)

[Web Based Bug Tracker](#)

[New Gitano Software Security Toolkit](#)

[CapeSoft Special Ends Sunday](#)

[Data Modeller New Release](#)

[PDF-XChange Update](#)

[PDF-XChange Evaluation SDK Now Available](#)

[SealSoft's xDigitalClock](#)

[Clarion SMS Service Survey](#)

[SealSoft xPictureBrowse 1.2 Released](#)

[Electronic Developer Network](#)

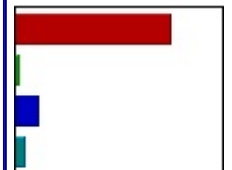
[Buggy 3.0.4 Available](#)

[RTF Tools Template Released](#)

[AppInit Version 2.0 Released](#)

SURVEY

Would a template dialog builder and debugger be useful to you?



- Yes (78%)
- No (I do write templates) (3%)
- No (I don't write templates) (12%)
- Don't know (5%)

155 Total Votes

[Previous Surveys](#)



Nice Touch Solutions, Inc.



Posted Monday, August 27, 2001

[QuickFlash Freeware
Template](#)

FAQ: Troubleshooting TPS File Corruption

In this free FAQ reprint, Eric Vail lists a number of questions and answers to help you resolve TPS file corruption over a network.

Posted Friday, August 24, 2001

Running Totals

Running totals come in two flavors. The first is the constant updating of an account balance in real time, such as a running total in a checkbook application. The other kind is, well, the subject of this 100th article by Steve Parker.

Posted Tuesday, August 21, 2001

Steve Parker's 100th!

Okay, so Dr. Parker is no spring chicken, but he's not that old. The 100 refers not to his age, but his output of Clarion programming information. This week, the publication of (appropriately enough) Running Totals marks Steve's 100th Clarion article!

Posted Tuesday, August 21, 2001

Weekly PDF for August 12-18, 2001

All Clarion Magazine articles for August 12-18, 2001 in PDF format.

Posted Monday, August 20, 2001

Migrating The Inventory Application To SQL Server (Part 2)

In this three part series, Ayo Ogundahunsi takes the Clarion example Inventory application from a flat file TPS database to a MS SQL database, focusing on portability, business rules, relational integrity, and Clarion as an interface tool.

Posted Monday, August 20, 2001

Frequently Asked Questions

Do you know how many times per year Clarion Magazine publishes? Ever wonder how many words of Clarion-related information there are on the ClarionMag web site? Do you want to know who's behind Clarion Magazine? Read on...

Posted Monday, August 20, 2001

Creating Filter Expressions The Easy Way

Creating filter expressions can be a tricky process - there's no "Populate Column" toolbox readily available and it's possible to make a typing error that's not picked until run-time. Here's a nifty workaround that can save some grief.

Posted Thursday, August 16, 2001

Must Be in This List

"Must Be in List" validation does little more than complete template prompts to call a lookup. But what if you need to validate against a specific list? Steve Parker shows how.

Posted Thursday, August 16, 2001

True Threading: What You'll Need To Know

Multitasking is an important part of Windows application development; we've all become used to running multiple applications at once, and most of us create MDI applications which can have multiple windows open, each on its own thread. Clarion does this sort of thing well, for the most part. So what's the problem, and why are developers pushing for "true" threading support in Clarion?

Posted Wednesday, August 15, 2001

Migrating The Inventory Application To SQL Server (Part 1)

In this three part series, Ayo Ogundahunsi takes the Clarion example Inventory application from a flat file TPS database to a MS SQL database, focusing on portability, business rules, relational integrity, and Clarion as an interface tool.

Posted Wednesday, August 15, 2001

Weekly PDF for August 5-11, 2001

All Clarion Magazine articles for August 5-11, 2001 in PDF format.

Posted Monday, August 13, 2001

Using Example Files With TPSFix

Sometimes TPS files continue to get corrupted even after you run TPSFIX. John Heck shows how to use example files to fix corrupted headers once and for all.

Posted Friday, August 10, 2001

Previous Surveys

Clarion Magazine has taken weekly reader surveys since January, 2001. On this page

you can view those survey results. You need to be a subscriber or a member to get access. [Membership](#) is free.

Posted Thursday, August 09, 2001

Internal Help - An Alternative To Commercial Help Systems

At its most basic, application Help is just a display of a specific block of text when a button is pressed. This is a lot like displaying the memo of a record upon demand, something that Clarion can do easily. That was the starting point for Tim Philips' integrated 'internal' help system.

Posted Thursday, August 09, 2001

Understanding Recursion - Part 2

What is recursion? Ask that question of any group of developers and chances are pretty good you will receive several different answers. Some of the answers will be correct and others will be...interesting. Part 2 of 2.

Posted Wednesday, August 08, 2001

Understanding Recursion - Part 1

What is recursion? Ask that question of any group of developers and chances are pretty good you will receive several different answers. Some of the answers will be correct and others will be...interesting.

Posted Friday, August 03, 2001

Using The Web Browser OCX

Have you ever wanted to display an HTML page from within your application without all the hard work and heartache of interfacing to Internet explorer? How about viewing and editing a document or spreadsheet without loading up Word or Excel? Well now you can! Ever since Internet Explorer 4 Microsoft has been supplying the Web Browser OCX, a wonderful little control which will do all this for you, and what's more it is incredibly easy to use!

Posted Thursday, August 02, 2001

A FileManager For Marked Deleted Records

Dennis Evans recently created a template and two classes to manage records marked as deleted (rather than physically deleted), as described in a recent ClarionMag article.

This page contains usage notes on the class, and a link to the source.

Posted Wednesday, August 01, 2001

Copyright © 1999-2001 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

Good help isn't that hard to find.

\$1.67 per issue

published by
CoveComm Inc.

Clarion MAGAZINE

Finding A File Using Template Code

by **Kevin Erskine**

Published 2001-08-31

As developers write their own Clarion classes either for their own use or as third-party add-ons, it becomes more difficult to only include the classes you want. Clarion requires all "ABC compliant" classes to be located in the LIBSRC directory. Class files include the class definition file (INC) and the method source file (CLW).

Clarion handles all (or most) of these class source files automatically, and while this makes life easier for the developer in many cases, it can also make things a lot more complicated for programmer's choosing to write their own templates and classes. To solve this problem I created my own functionality for including class source files in the IDE in a more controlled manner, and that meant I had to write template code to find the required files. But to explain all of that, I need to go back to how the Clarion IDE handles ABC classes.

An ABC compliant class is one where one of the first lines in the include file reads:

```
!ABCIncludeFile
```

This line tells the Clarion IDE to automatically parse the include file and add any defined classes to its list of ABC classes. When you create a multi-DLL application the compiler will compile and link in all these classes automatically. This causes two problems;

1. Application bloat – the compiler links in a whole lot of code that may not ever be used by any related DLLs. Since the "data owning" DLL does not know what any dependent DLLs will need it includes everything.
2. Missing DLLs – due to (1) above, additional DLLs may be needed for deployment. You may have to include unwanted database drivers or even other third-party DLLs for your application to run on a client's machine.

All the classes I write are ABC, they are just not ABC compliant (in Clarion's eyes). Since I always write a template wrapper for my classes this is not an issue; the template will make sure the proper

[Search](#)[Home](#)[COL Archives](#)[Information](#)[Log In](#)[Membership/](#)[Subscriptions](#)[FAQ](#)[Privacy Policy](#)[Contact Us](#)[Downloads](#)[PDFs](#)[Freebies](#)[Open Source](#)[Site Index](#)[Call for](#)[Articles](#)[Reader](#)[Comments](#)

source files and embed points are included in the application automatically.

In order to support this feature I needed to write a template that would define and export my class structures correctly. For this to happen automatically, the template had to read the include file and create the proper export statements for the DLL's export list. The problem was that not everyone installs their classes in the LIBSRC directory, so the template might have to do some searching. Hence, the need for some "Find File" template code. Please note that I am not describing my export template in this article (maybe in another one); just the file finder code is discussed below.

Finding a file in template code

The following code defines the %FileToFind template symbol and then calls the template function (a #GROUP) called %FindFileOnDisk. If the function doesn't find the file, the template displays an error message in the compile log and terminates the compile.

```
#EQUATE(%FileToFind, 'MyClass.INC')
#IF(NOT CALL(%FindFileOnDisk,%FileToFind))
    #ERROR('Export Class Error: File Not Found: ' <←
        & %FileToFind)
    #ABORT
#ENDIF
```

The %FindFileOnDisk #GROUP

Clarion uses the Redirection (RED) file to locate files it needs to compile and link your application, and I took the same approach. I won't go into detail on the structure and use of the Redirection file. If you would like further information please consult Clarion's documentation.

Clarion's Redirection file uses the same format as Window's INI file. It has Sections and Entries and thus can be read using the GETINI function provided by the Clarion language.

Step 1 is to read the Window's INI file to locate information about where Clarion is installed on the development machine. The template then builds the name of Clarion's Redirection file:

```
#!===
#!=== These 2 Lines Dependant on CW version
#!=== Not all are listed
#!===
#!=== 1) Read the WIN.INI file to find out
#!=== where Clarion is installed
#!=== 2) Now Set the name of Clarion's
#!=== Redirection File
#!===
```

```
#CASE (%CWTemplateVersion)
#OF ('v5.5')
  #SET (%FindCWRoot, GETINI('Clarion 5.5
    Enterprise Edition','Root'))
  #SET (%FindCWRED , %FindCWRoot & 'BIN\C55EE.RED')
  #SET (%FindCWIni , %FindCWRoot & 'BIN\C55EE.INI')
#ENDCASE
```

Next, the template parses out the extension of the filename to be located and looks for that extension in the [COMMON] section of the Redirection file.

```
#!===
#!=== Read the RED file (It is in INI format)
#!=== I only use [Common] settings.
#!=== Look for the Entry of the file
#!=== pattern that was passed in. (e.g.
#!=== *.inc, *.gif, *.clw, etc..)
#!===
#SET (%FindPos, INSTRING('.',%FileToFind,1,1))
#IF (%FindPos)
  #SET(%FindTemp,'*' & SUB(%FileToFind,%FindPos))
  #SET(%FindRedLine,GETINI('Common',%FindTemp,'',
    %FindCWRED))
#ENDIF
```

There should also be an entry of *.* which Clarion will use for extensions that don't match the extension pattern. This entry is used to find any files not having an extension pattern in the redirection file. The template reads this entry and appends it to the list of search directory locations. Since a semicolon separates each directory entry, the template code below adds an extra semicolon to the end of the symbol for the parsing loop to work correctly.

```
#!===
#!=== Also Append the value for all
#!=== files the *.*= entry
#!===
#IF (%FindTemp NOT = '*.*')
  #SET (%FindTemp ,GETINI('Common','*.*',
    '',%FindCWRED))
#ENDIF
#!===
#!=== Tack on an extra ';' for below loop
#!=== code to work correctly
#!===
#SET (%FindRedLine ,%FindRedLine & ';' &
  %FindTemp & ';')
#!===
```

Clarion also has something called "Redirection Macros" that can be used in the redirection definitions to help globalize certain values. One of these used is called %ROOT% and is set to Clarion's location. The programmer can define additional macros in the Clarion INI

file (e.g. C5EE.INI) under a section called "Redirection Marcos." See the Clarion reference manual for information on how to do this. Below is a call to a function called %ReplaceText, which will replace all the occurrences of any macros found. I'll have more to say about %ReplaceText below.

```

#!===
#!=== If the Programmer did not override %ROOT%
#!=== in the Clarion INI Redirection Macros
#!=== Section, use the value from the Windows INI
#!===
#SET(%FindPathName,GETINI('Redirection Marcos',←
    'ROOT','','%FindCWIni))
#IF (%FindPathName = '')
    #CALL(%ReplaceText,%FindRedLine, '%ROOT%',←
        %FindCWRoot)
#ENDIF
#!===
#!=== See if any other [Redirection Macros] are used
#!===
#SET(%FindPos, INSTRING('%','%FindRedLine,1,1))
#LOOP WHILE(%FindPos)
    #SET(%FindPos, %FindPos + 1) #! Skip %
    #SET(%FindTemp, SUB(%FindRedLine,%FindPos,(INSTRING←
        ('%','%FindRedLine,1,%FindPos + 1) - %FindPos))
    #SET(%FindPathName,GETINI('Redirection Marcos',←
        %FindTemp,'','%FindCWIni))
    #SET(%FindPathName,LEFT(%FindPathName))
    #SET(%FindPathName,CLIP(%FindPathName))
    #IF (SUB(%FindPathName,-1,1) NOT = '\')
        #SET(%FindPathName,%FindPathName & '\')
    #ENDIF
    #CALL(%ReplaceText ,%FindRedLine, ('%' & %FindTemp ←
        & '%'), UPPER(%FindPathName))
    #SET(%FindPos, INSTRING('%','%FindRedLine,1,1))
#ENDLOOP

```

Begin by setting the find flag to %FALSE and begin a #LOOP:

```

#!===
#!=== For every Directory Specified in the
#!=== RED line Check if File Found
#!===
#SET(%FindReturnCode,%FALSE)
#LOOP

```

Parse out the next directory to search. This is everything in front of the semicolon in the redirection string:

```

#!===
#!=== Get semicolon separator for
#!=== next directory to search
#!===
#SET(%FindPos, INSTRING(';','%FindRedLine,1,1))
#IF (%FindPos = 0)

```

```

        #! If no more found then we are done searching
        #BREAK
#ENDIF
#!===
#!=== Parse this information out, and add
#!=== trailing '\' if not already there
#!===
#SET(%FindTemp,SUB(%FindRedLine,1,%FindPos - 1 ))
#IF (SLICE(%FindTemp,LEN(%FindTemp),←
        LEN(%FindTemp)) NOT = '\')
        #SET(%FindTemp,%FindTemp & '\')
#ENDIF

```

Now build a path plus filename specification to search using this information:

```

#!===
#!=== Build the file specification
#!=== we are looking for w/ this directory
#!===
#SET(%FindTemp,%FindTemp & %FileToFind)

```

The Clarion template language also provides the programmer with a function to easily see if a file exists on your disk. The template uses this function, and if the file is found it updates the original filename symbol passed into the function with the complete LONGPATH name of the file's location. It also sets the find flag to %TRUE to let the calling code know the function was successful and #BREAK out of the #LOOP.

```

#!===
#!=== Use the FILEEXISTS function to see if
#!=== we found the file. If found SET the
#!=== return field with the full filename
#!===
#IF(FILEEXISTS(%FindTemp))
        #SET(%FileToFind      , %FindTemp)
        #SET(%FindReturnCode,%TRUE      )
        #BREAK
#ENDIF

```

If the file was not found, remove the last directory from the front of the redirection line and LOOP again.

```

        #!===
        #!=== If not found then remove the directory
        #!=== just search from the front of the
        #!=== Redirection Line and cycle again
        #!===
        #SET(%FindRedLine,SUB(%FindRedLine,%FindPos + 1))
#ENDLOOP

```

Finally, the template returns the status of the search to the calling code:

```
#!===
#RETURN %FindReturnCode
#!===
```

Definition of %ReplaceText:

The following is a simple function to take a string symbol (OrigText) and replace all occurrences of "OldText" with "NewText". It is used in the above code to replace the redirection macros with the actual path information found in the Clarion INI file.

```
#!===
#!=== Substitute NewText for OldText in
#!=== OrigText -- All occurances
#!===
#DECLARE(%StringPos)
#LOOP
  #SET(%StringPos, INSTRING(UPPER(%OldText), ←
    UPPER(%OrigText), 1, 1))
  #IF(%StringPos = 0)
    #BREAK
  #ENDIF
  #SET(%OrigText, SUB(%OrigText, 1, %StringPos - 1) ←
    & %NewText & SUB(%OrigText, %StringPos + ←
    LEN(%OldText) + 1))
#ENDLOOP
#!===
```

To include this template code into one of your own template chains simply download the source (at the end of this article), and code the following into the chain's TPL file:

```
#INCLUDE( 'SBR_FIND.TPW' )
```

I use this template code in several of my other template sets, including my class management templates. You can use this template code to locate and parse files wherever your templates need to extract information and generate conditional code.

[Download the source](#)

[Kevin Erskine](#) lives in Sausalito, California (USA) where he works from his home and enjoys life immensely. He is the owner of [Software By Ragazzi](#), which specializes in complete software solutions from conception through deployment. He has been using Clarion since the original DOS version. From native TPS systems, to MS SQL and upcoming XCOM and ADO features, Kevin currently does not see any problem Clarion cannot solve today or in the near future.

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 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

In The Red

by Andrew Guidroz II

Published 2001-08-29

In accounting, it's standard practice to show negative numbers in red. Here's a source procedure that will search the currently active window for any numeric fields capable of displaying negative numbers. If the procedure finds such a field with a negative value, it sets the text color to red; otherwise it restores the default color.

This procedure exploits a scoping peculiarity of Clarion. No matter which procedure you are in, your code has access to the last window opened on that thread. You don't need a reference to the current window; you can simply use those Clarion language statements that act on the current window. `DisplayNegativesInRed` uses the `FirstField()` and `LastField()` functions to get the minimum and maximum field equate numbers on the current window. It then uses the property syntax to retrieve the field's picture (`PROP:TEXT`) and contents (via the use variable, `PROP:USE`), and then sets the color (`PROP:FontColor`) based on the contents. Using `COLOR:NONE` sets the field's color back to the default.

To use this procedure in your application, create a new Source procedure called `DisplayNegativesInRed`. Copy the two local variables into the data embed, and everything below the `CODE` statement in to the code embed. This procedure doesn't take any parameters or return any value.

```
DisplayNegativesInRed PROCEDURE
Loc:CurrentControl   SIGNED
Loc:HoldText        STRING(255)
CODE
Loop Loc:CurrentControl = FirstField() to LastField()
  IF Loc:CurrentControl{Prop:Type} = Create:SString |
    OR Loc:CurrentControl{PROP:Type} = Create:Entry
    Loc:HoldText = Loc:CurrentControl{Prop:Text}
    IF UPPER(Loc:HoldText [ 1 : 2 ]) = 'N-'
      IF Loc:CurrentControl{PROP:Use} < 0
        IF Loc:CurrentControl{PROP:FontColor} <> Color:Red
          Loc:CurrentControl{PROP:FontColor} = Color:Red
        END
      ELSIF Loc:CurrentControl{Prop:FontColor} = Color:Red
        Loc:CurrentControl{PROP:FontColor} = Color:None
      END
    END
  END
END
```

[Search](#)

[Home](#)

[COL Archives](#)

[Information](#)

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)



END

END

END

Andrew Guidroz II, when he isn't handfeeding the tufted titmouse, writes software for all facets of the insurance industry. His famous Cajun cookouts have become a central feature of Clarion conferences throughout the U.S. Andrew's Cajun website is www.coonass.com.

Reader Comments

[Add a comment](#)

Color switcher could protect existing colors by only...
Nice additions, Carl. Thanks a bunch.

Copyright © 1999-2001 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.

INVEST
in your own abilities

Clarion
magazine

published by
CoveComm Inc.

Clarion MAGAZINE

Migrating The Inventory Application To SQL Server (Part 3)

by **Ayo Ogundahunsi**

Published 2001-08-28

[Last week](#) I showed how default values can be made an integral part of your tables at creation. You can also add default values later. Used carefully, defaults can help standardize constants especially when maintaining the bridge between Clarion and MSSQL.

Date discrepancies

A common example is the use of defaults in preventing the "Record has been changed by another station" error that occurs even on a single machine sometimes, and has been frustrating to a lot of users that have posted questions on the SoftVelocity newsgroups.

In the earlier discussion about default values I mentioned that even though the Clarion Synchronizer is smart enough to convert the `TODAY()` function to its SQL equivalent `GETDATE()`, this causes problems later. This is what happens:

There are two Date data types in SQL Server, namely `SMALLDATETIME` and `DATETIME`. `SMALLDATETIME` has accuracy up to one minute. e.g. 2001-05-03 03:57:00.000, while `DATETIME` has accuracy up to 3.33 milliseconds, e.g. 2001-05-03 03:57:26.480.

The problem with a `GETDATE()` default value is that if for any reason rows are added to your database outside Clarion, for example from a stored procedure, the column will be updated with a value of a higher precision (as indicated above) because `GETDATE()` returns the current date and time as a `DATETIME` data type. So, later, when a user tries to edit that row from a Clarion Update Form, she gets this message:

"This record was changed by another station. Those changes will now be displayed. Use the Ditto button or Ctrl+ to recall your changes"

[Search](#)

[Home](#)

[COL Archives](#)

[Information](#)

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)



The error occurs because Clarion is unable to handle the precision of milliseconds. In order to resolve this, you can change the data type on the backend from DATETIME to SMALLDATETIME if you do not need to deal with seconds. In most cases, this is unacceptable. The other option is to create a kind of global default in SQL Server that controls the precision of date values whenever updated. Fortunately, it is possible to populate a database with default values by the backend, no matter where the update is coming from, i.e. from within Clarion, or from a stored procedure. To do this, I will be making use of the following Transact-SQL functions:

DATEPART() - Return an integer value which represents the part used.

CAST() - Convert data from one data type to another.

CONVERT() - Similar to CAST() except with additional date formatting.

You can find a detailed explanation of these functions in the SQL Server's Books online.

What I want to do is to split the value returned by GETDATE() into parts, and rebuild the date, but this time with zeroes as milliseconds. Any default values will then be Clarion-compatible. Here's the code:

```
SELECT CONVERT(datetime,
(
CAST(DATEPART(yyyy, GETDATE()) AS varchar) + '-' +
CAST(DATEPART(mm, GETDATE()) AS varchar) + '-' +
CAST(DATEPART(dd, GETDATE()) AS varchar) + ' ' +
CAST(DATEPART(hh, GETDATE()) AS varchar) + ':' +
CAST(DATEPART(mi, GETDATE()) AS varchar) + ':' +
CAST(DATEPART(ss, GETDATE()) AS varchar)
)
,120)
```

Notice that I put a SELECT before the actual conversion. Get used to this; it's a commonly used way to get the result of expressions in SQL.

Now, copy this and run in the Query Analyzer, and you will get something like:

```
2001-05-03 03:57:26.000
```

depending on your time.

The Transact-SQL script to create a user defined Default called TODAY that I can use globally to populate any column is this:

```
CREATE DEFAULT [TODAY] AS
CONVERT(datetime,
```



```
(
CAST(DATEPART(yyyy, GETDATE()) AS varchar) + '-' +
CAST(DATEPART(mm, GETDATE()) AS varchar) + '-' +
CAST(DATEPART(dd, GETDATE()) AS varchar) + ' ' +
CAST(DATEPART(hh, GETDATE()) AS varchar) + ':' +
CAST(DATEPART(mi, GETDATE()) AS varchar) + ':' +
CAST(DATEPART(ss, GETDATE()) AS varchar)
),120)
GO
```

Cut, paste, and execute this snippet in your Query analyzer, and your default will be created. Note that the default can also be created from the Enterprise manager. However, it is better you use a script to create this since it is more difficult to enter a formula in a one line prompt. Later, you will see how to use this DEFAULT in your tables.

Running scripts

The synchronizer generates the SQL table creation script from the definitions in your dictionary. You should load this script into your Query Analyzer. As you do this, you have to remember to change your active database from "master" to INV_SQL or else, the tables will be created inside the master database. You really don't want to do that.

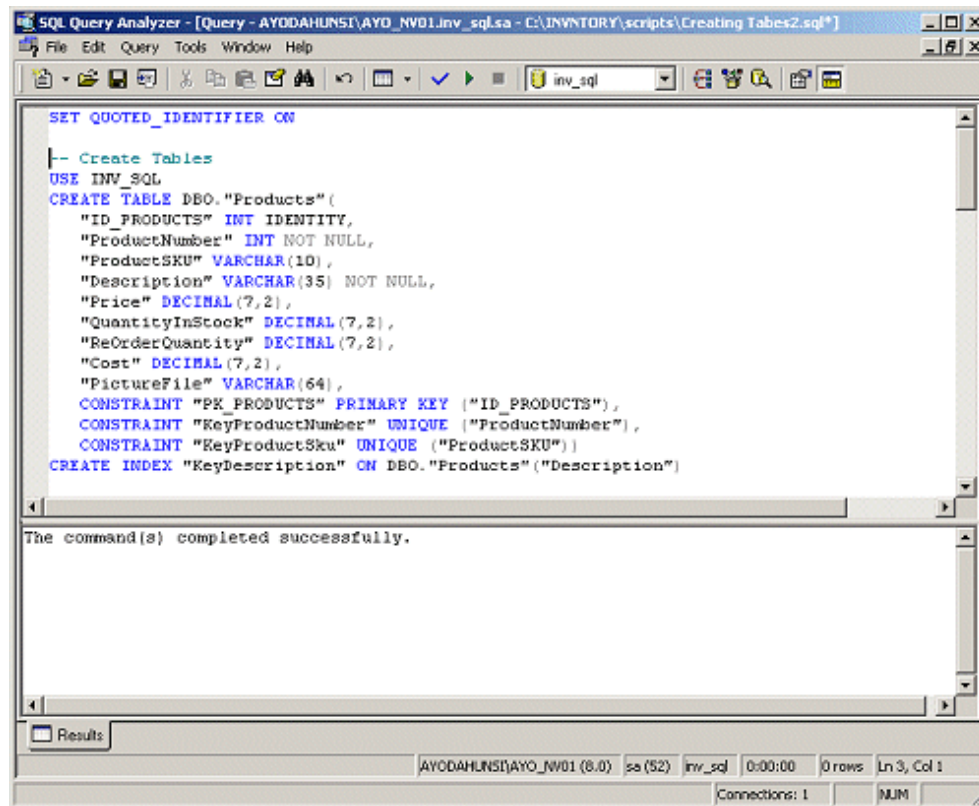


Figure 10. Running script to create tables.

Now you can run the script. After the script has generated the necessary tables and indexes, constraints, etc., you need to change the default value of GETDATE() to dbo.TODAY using the

global default called `TODAY` you just created. This will help prevent precision errors discussed earlier.

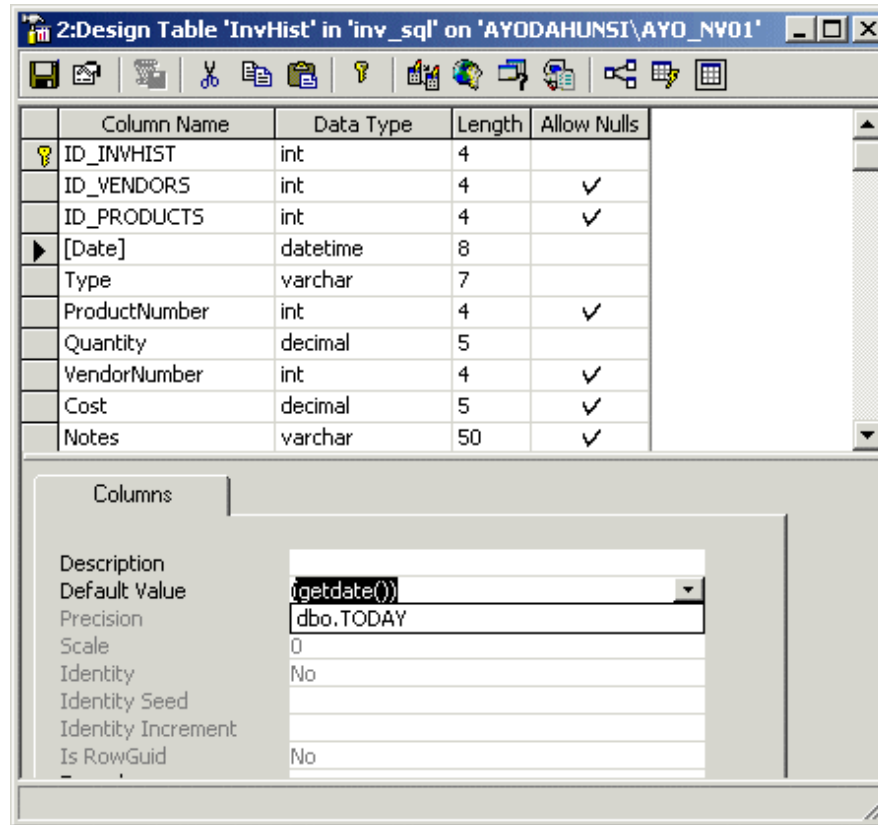


Figure 11. Setting Date defaults as `dbo.TODAY`.

You get to the window shown in Figure 11 by right-clicking on the table and selecting Design Table from the context menu.

Data conversion and the role of constraints

If you take a look at the script created with the synchronizer, you will see some `CONSTRAINT` statements towards the end of the script. You can read a detailed explanation on constraints and referential integrity in an earlier series [articles on SQL](#) by Dave Harms. Nevertheless, as a recap, a primary key constraint will ensure that all rows (records) in a table will have a unique key, and a foreign key constraint will ensure that a relationship exists between two tables.

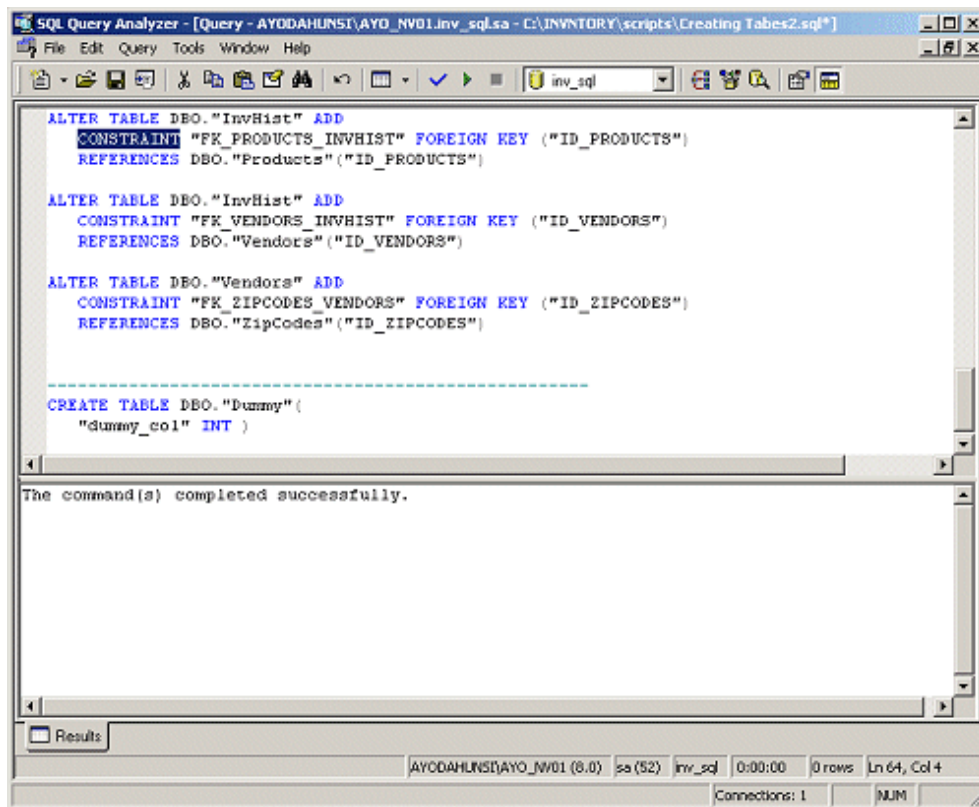


Figure 12. Constraint statements in SQL script.

For example, you cannot have a record inserted into the VENDORS tables without an ID from the Zip Code table. Now this presents a problem considering the way Clarion inserts a blank record when it tries to auto-increment; when you try to insert a record into the VENDORS table, Clarion will actually try to add a blank record first. Unfortunately, when foreign key constraints are used, SQL Server will not allow this since the column - ID_ZIPCODES in the VENDORS table has not yet been filled with any value.

According to the Clarion help on Relationship Properties in the Dictionary, setting the Referential Integrity Constraints to Restrict, Restrict (Server), Clear (Server) are supposed to generate different kinds of code within the SQL script. However, from what I have seen so far, whatever setting you use will still generate these Foreign Key constraints.

For now, you can remove these foreign constraints. You can do this through the Enterprise Manager or by running the code script show below:

```

ALTER TABLE DBO."InvHist"
DROP CONSTRAINT "FK_PRODUCTS_INVHIST"
ALTER TABLE DBO."InvHist"
DROP CONSTRAINT "FK_VENDORS_INVHIST"
ALTER TABLE DBO."Vendors"
DROP CONSTRAINT "FK_ZIPCODES_VENDORS"

```

The application

Remember that you deleted all the procedures in the application; you now have to import the procedures again from the original example application – INVNTORY.APP. When you do this Clarion will bring up an error about the dictionaries being different. This is no cause for concern since you didn't change any fields, rather, you only added some new ones. Make sure you import all the procedures except the `MainFrame` because you have already added the assignment to the connection string in a `MainFrame` embed point.

The application should compile fine without any errors. However, If you attempt to run the converted application, you are likely to get this error message:

File (DBO.Vendors) could not be opened. Error: ODBC.DLL Could Not Be Loaded (1). Press OK to end this application.

This happens when you compile your EXE as a 16 bit application, which is the default mode for the shipped Inventory Application in versions prior to 5.5. Change your application to 32 bit and this is resolved.

When working with SQL-based systems, you need to start thinking of the files (tables), stored procedure, and views as one single unit called the database. So, it is better to open all the files (tables) at the start of the application, and close them when exiting. Of course, in applications that have a lot of tables, this can take a long time, but once the tables are opened, whatever procedure needs to use these tables will not have to reopen them again. Note also that an attempt to open the first table establishes a connection (using your connection string) with the database.

After you have made the changes to the example application, running the application and exiting will give you the GPF shown in Figure 13:

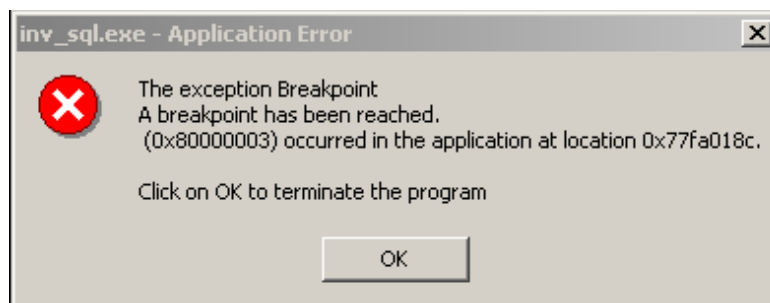


Figure 13. A breakpoint exception exiting the SQL application.

You can resolve this GPF by adding the `INVHIST`, `PRODUCTS`, `VENDORS`, `ZIPCODES` tables to the `OTHER TABLES` section of the `File Schematic` in the `MainFrame` Procedure. By doing this, the tables are opened as the `MainFrame` is opened and closed properly when the frame closes.

Which way to go

Having gone through the basic conversion of a Topspeed file system to a SQL Server system, you need to ask yourself if you want to stop here and deploy the application to your customers in the present state. Or do you want to move out the underlying business logic to the server as stored procedures and database constraints? For instance, it would be nice to have a trigger that generates an email message when the stock quantity of any product falls below the re-order level.

As I said before, how much code to move to the server is a business decision. One thing is certain: SQL systems are rapidly replacing flat-file systems, and programming languages/interfaces are increasingly transferring business logic to the backend.

[Download the source](#)

[Ayo Ogundahunsi](#) presently lives in Henderson, Nevada, about ten minutes from Las Vegas. He works for [Impac Medical Systems Inc.](#), the leading company in cancer therapy software (written in Clarion). Impac has its headquarters in Mountain View, California. Ayo is married to Ayodola, and they have two boys, Darren and Joshua.

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 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

Subscribe to Clarion Magazine

Published 2001-08-27

This page is for **online purchases**. For information on check and bank transfer payments, or if you have any other questions, please contact subscriptions@clarionmag.com.

Choose a subscription/renewal option:

\$80 - One year

\$150 - Two years

Clarion Magazine handles all purchases through the following vendors. Accordingly, your credit card statement will show the amount of your subscription paid to the vendor you choose, rather than to Clarion Magazine.

Choose a payment method

Credit card online via DeveloperPlus

Your credit card statement will show the amount of your subscription paid to **DeveloperPlus Corporation**.

Credit card by phone or fax

You will be sent to a page describing where to call/fax in your order.

Check, money order, or bank transfer

You will be sent to a page describing now to make your payment. Checks must be drawn on a US or Canadian bank, in US funds.

IMPORTANT

By continuing with this order you are indicating your agreement to the terms of the Clarion Magazine [subscription agreement](#).

Online payments are normally processed a few minutes, but may take up to one business day.

[Search](#)

[Home](#)

[COL Archives](#)

[Information](#)

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)



Reader Comments

[Add a comment](#)

Copyright © 1999-2001 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Clarion MAGAZINE

Clarion News

[SealSoft xQuickFilter v2.0.4 Released](#)

New in xQuickFilter 2.0.4 : default use of UPPER() on all fields in filter, can be turned off; new method to check if QuickFilter is on or off; minor cosmetic changes. A new install and demo are available.

Posted Friday, August 31, 2001

[Web Based Bug Tracker](#)

Bugtrack is a web based bug tracking system written in Perl/DBI.

Posted Friday, August 31, 2001

[New Gitano Software Security Toolkit](#)

Gitano Software has introduced G-Sec, a password protection toolkit. You can drop in the global template, or use any of several extension or code templates to add more control. You can also use the included procedures to manage user access, or write your own. Demo and registered version will be available for download September 1, 2001. Regular price is \$149, introductory price until September 15th is \$99. Competitive upgrade and bundle available, as is an upgrade from GReg. GSec is distributed as a template/dll solution, available for C5 and C55, 16 and 32 bit, stand alone compiles only.

Posted Friday, August 31, 2001

[CapeSoft Special Ends Sunday](#)

Just a reminder to all that CapeSoft MessageBox and CapeSoft HyperActive will be reverting to their normal prices from Monday morning on. Message Box is currently on special for \$39 (down from \$49) and HyperActive is at \$29 down from \$39.

Posted Friday, August 31, 2001

Data Modeller New Release

A new version of the Clarion Data Modeller is now available. Before installing please delete all other copies of dm5.exe and dm55.exe. For more information email peabrain@global.co.za.

Posted Thursday, August 30, 2001

PDF-XChange Update

An update is available to all owners of the PDF-XChange SDK. The download is 1.6MB and is password protected. Please note the drivers are no longer included as these are in a self-contained install. This update includes new templates and updated classes (big thanks to Craig Ransom) and it is important to read the help first, particularly regarding the driver template.

Posted Thursday, August 30, 2001

PDF-XChange Evaluation SDK Now Available

You can now download a fully functional version of the PDF-XChange SDK for Clarion including templates, classes, API, drivers, demo app and manual. The limitation is that each page created by the drivers contains a watermark. You will need a password to access the install so please send an email to Support@docu-track.com with a subject tag or message stating something like "Clarion PDFX SDK password." There is also a free [developer support forum](#) now available:

Posted Thursday, August 30, 2001

SealSoft's xDigitalClock

SealSoft's xDigitalClock is a simple class and control template which displays time like digital Clock. Digital skins are supported. Future releases will add timer, stopwatch, and time zone capabilities. An xDigitalClock install for Clarion 5/5.5 will be available shortly.

Posted Tuesday, August 28, 2001

Clarion SMS Service Survey

Cliff Court is considering offering Clarion developers an SMS (Short Message Service) in most areas where GSM cell networks exist. Clarion developers would be able to use it by either emailing or sending via a Microsoft Message Queue message. Cliff's company has direct links into a major GSM network that can deliver to most countries around the world (sorry that excludes the USA due to the limited use of GSM there so far). But for others in Europe, Oz, South Africa and the Far East, this may be a useful service. So please let Cliff know if it sounds viable to you. The service would entail a monthly minimum fee of around US\$50, including a minimum of 500 SMS messages, thereafter US\$0.10 per message (not \$0.80 as earlier reported - ed.). For those with very high volume, discounts can be negotiated. Please note this could not be used as a SPAM channel.

Posted Monday, August 27, 2001

SealSoft xPictureBrowse 1.2 Released

New in SealSoft's xPictureBrowse v1.2: assign a variable for the default open directory; assign different variables for each instance of xPictureBrowse; save settings in INI files; minor cosmetic improvements. A new install and demo are available.

Posted Monday, August 27, 2001

Electronic Developer Network

eDN (Electronic Developer Network) is a free online resource for Clarion Developers. This site includes a knowledge base, download section, newsgroups, and news items. The site is intended to help beginners get started with SoftVelocity software. Help is available mainly for beginner-level issues.

Posted Monday, August 27, 2001

Buggy 3.0.4 Available

An update to the Buggy bug tracking tool is now available to all registered users. The demo has also been updated.

Posted Monday, August 27, 2001

RTF Tools Template Released

RTF Tools is a Clarion template that allows a developer to implement the following measures into Applications created with the ABC template chain: print RTF control contents as standard text on a Clarion report; convert Softvelocity RTF control contents to HTML; create an HTML file from RTF control contents. This template adds two classes to the LIBSRC directory and is not implemented as a LIB or DLL. This means that the developers can modify the classes as needed. This template currently works only with the RTF control implementation from SoftVelocity. Demo available.

Posted Friday, August 24, 2001

AppInit Version 2.0 Released

Version 2.0 of AppInit is now available for download. The setup program is password protected and there is a new password for version 2.0. If you have purchased version 1.0 you are entitled to an upgrade to all future versions of AppInit for Clarion 5/5.5 at no extra charge. You can request your new password by email. New features in 2.0 include: an extension template specifically for DLLs, which will only include the version stamping portion of AppInit and not the Splash Screen or About Screen functions; disable exporting of ABC compliant libraries that you are not using in your application; About window can list your application DLLs and version information and mark out of date DLLs; limit the application to a single instance (using the CreateMutex API call); obey a Windows shutdown request; new method of stamping version info into the code (no global data used); translatable strings; disabling splash screen at design or runtime; option to disable AppInit code generation without removing the template.

Posted Thursday, August 23, 2001

QuickFlash Freeware Template

In early 2001 Mike McLoughlin started work on a template to export all Quicken Quickbooks. Subsequently, Quicken announced they were going to bring out an XML interface to replace the IIF files that QuickFlash produces. As a result, Mike has stopped work on the IIF files and is now releasing as freeware the QuickFlash template in its current unfinished state, so that it may serve as a teaching example for anyone who still needs/wants to

create IIF files.

Posted Tuesday, August 21, 2001

Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the expresswritten consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

INVEST
in your own abilities

Clarion
magazine

published by
CoveComm Inc.

Clarion MAGAZINE

FAQ: Troubleshooting TPS File Corruption

Published 2001-08-24

Although TPS files are generally quite reliable, every once in a while someone posts an urgent message in the SoftVelocity newsgroups about TPS file corruption on a network. [Eric Vail](#) recently posted the following list of questions and answers, reprinted here with Eric's permission. For additional resources, see Mark Riffey's [network checklist](#).

Questions to ask yourself

- Is your program the only application sharing data over the LAN?
- When did data corruption begin?
- What changed in the LAN at that point?
- Was there a new program loaded or some new hardware introduced?
- Is there enough open space on the server where the data is stored?
- When is the last time the cooling fans were checked on the processors in the server and or the case for that matter? When you are running a database the server processors run hotter than usual so if the fans are failing it can cause the server to mess up.
- Does the server ever lock up? What about the workstations? This could be caused by a flaky power supply and/or RAM.
- When was the last time the server or workstation disks were defragged? (NT does NOT defrag on the fly like Novell.) There are some utilities out there but usually we partition the drives C and D and put the boot and system stuff on C and all programs and data on D. This way when we do maintenance we can just copy away the D data somewhere, then reformat the drive and do a scandisk before copying back the programs and data. On W2k you can defrag the partitions in

[Search](#)

[Home](#)

[COL Archives](#)

[Information](#)

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)



the computer manager.

Verifying network integrity

If none of the above solves the problem, then I would start looking at network hardware, specifically the network card and cables for the server. Here is an easy way to verify network integrity.

Get a block of data that you can verify size and number of files on. I usually use the I386 directory from an NT CD. It is large and has lots of big and little files in it so you get a good sampling of all data types and sizes.

Now put that directory onto the server somewhere and then copy it from the drive on the server in the following sequence.

- a. Copy from the server to each station. Time it and see how long it takes, then make sure that *all* the data makes it. Do a right click properties on the folder and make sure the number of files matches the server. Also make sure the total size of the files matches - *not* the space the files take up on disk, but the *total file size*, which is the number above the total disk space. That is the true size of the files.
- b. Next do the same from station to station.

If you find that one station-to-station copy is slower than another, or some files were dropped, move the cable in the hub first and make sure that the hub is okay. I have seen where one port in a hub will go bad and cause all kinds of flaky problems. The problem could also be a loose cable or a bad network card.

If all server-to-workstation and workstation-to-workstation copies are the same then you need to go back to the server and check the following:

- a. How much RAM is in the server and what type. The problem may be memory going bad.
- b. What type of hard drives are in the server and what configuration. If they are mirrored are they healthy? If RAID 5 are they optimal?
- c. Is the swap file size adequate? It should be twice the size of the physical RAM if you are sharing data files on it. That is not the default setting by the way. So if they have 256 meg of RAM then it should be 512 meg minimum, with a 768 meg ceiling.

Reader Comments

[Add a comment](#)

A good way to time the copy is to use a Batch file like...
Don't forget Windows 95 - ok it's not so common now to
have...
You've left out the 2 most common reasons for TPS...

Copyright © 1999-2001 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

Running Totals

by **Steven Parker**

Published 2001-08-21

(Editor's note: This is Steve Parker's 100th Clarion article; a complete bibliography is available [here](#).)

Running totals come in two flavors (there may be more but I am only aware of two). The first that comes to mind, mine at least, is the constant updating of an account balance in real time. The quintessential example of this type of running total is a checkbook where the account balance is updated as each check or deposit is entered or updated.

I ran into the second kind of running balance working in the International Department of a bank. The internal accounting group booked the FX (foreign exchange) transactions. With a few exceptions, none of these bookings were current transactions. That is, they were usually valued (dated) one to three days in the future. Some automatically unwound (e.g., the bank sold dollars and purchased sterling tomorrow, and sold the sterling back for dollars for the day after); some did not. Some were weeks in the future.

The report produced showed the balance of each of the due-from (vostro) accounts (bank accounts in another country) *as of each transaction*. In this way the FX traders could see their position forward. This report might show that as of today, if nothing further was done, the bank's d-mark account was going to be overdrawn several million marks in 10 days or that there would be a pile of swissy in four days which did no good just sitting in a bank in Geneva.

This report was called the "Position Report" and, as a result, I've taken to calling this type of running balance as "positional." This is the type of balancing act that Quicken does (though I fail to see the utility of this sort of balance information when balancing a personal checkbook).

There are probably several ways of doing each of these kinds of running balance in template-based procedures. I won't pretend to determine which type is best for your needs but I will show you at

[Search](#)

[Home](#)

[COL Archives](#)

Information

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

Downloads

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)

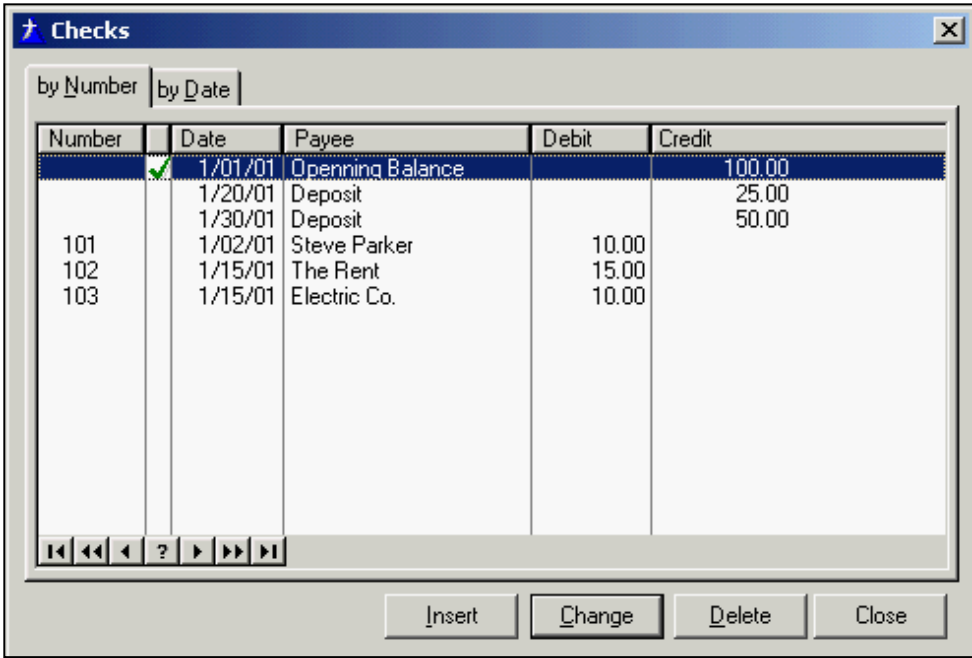


least one way of doing each.

Preliminaries

I do not store debit and credit amounts separately at the transaction level (though as I deal with accounts receivable software more and more, I find myself wishing this were considered good practice). Some applications I have seen store a sign and use that sign to determine how to handle an amount. Other applications store a separate debit/credit mark (my own checkbook application uses a separate, single character string to flag debit/credit).

While amounts may be stored in a single field, I expect a browse of an account to show debits on the left and credits on the right as, in Figure 1.



Number	Date	Payee	Debit	Credit
	1/01/01	Opening Balance		100.00
	1/20/01	Deposit		25.00
	1/30/01	Deposit		50.00
101	1/02/01	Steve Parker	10.00	
102	1/15/01	The Rent	15.00	
103	1/15/01	Electric Co.	10.00	

Figure 1. "Proper" display of debit and credits

In fact, the fields displayed here are check number/transaction reference, a reconciled mark, date, payee/description, debit amount and credit amount.

To properly reformat a single amount field into two fields as shown, I use two local fields in the browse box, creatively labeled "Debit_" and "Credit_." The following code in SetQueueRecord, before Parent call formats debits and credits:

```
Case CHE:DRCR
Of 'D'
  Debit_ = CHE:AMOUNT
  Credit_ = 0
Of 'C'
  Credit_ = CHE:AMOUNT
  Debit_ = 0
```

End

Because `Debit_` and `Credit_` both have the "blank when zero" attribute, I get a very readable display.

Had I used a sign, I suspect my code would have looked like this:

```
If CHE:Amount < 0
  Debit_ = ABS(CHE:AMOUNT) !don't show minus sign
  Credit_ = 0
Else
  Credit_ = CHE:Amount
  Debit_ = 0
End
```

Ok, now you know how I structure my apps and the generic data structures and window layouts I prefer.

Positional balances

The templates have simple totaling built-in. Unfortunately, this feature does not seem to support positional balances (see the sample in the attached app).

Because a positional balance must show the balance as of each record, an additional field is required. I use `R_Balance` for the purpose and populate it in the list box immediately to the right of the credit column.

`R_Balance`, like `Debit_` and `Credit_` is a local variable, not a file field. If it were a file field and a record were added in the middle of the sort order, all records would have to be recalculated *and* rewritten. With a local variable, as painful as it may be, only a recalculation is needed and that recalculation can be done entirely in memory.

The following modification to the debit/credit display code ensures that `R_Balance` is correctly calculated with each transaction record displayed:

```
Case CHE:DRCR
Of 'D'
  Debit_ = CHE:AMOUNT
  Credit_ = 0
  R_Balance -= CHE:AMOUNT
Of 'C'
  Credit_ = CHE:AMOUNT
  Debit_ = 0
  R_Balance += CHE:AMOUNT
End
```

Figure 2 shows the result.

Number	Date	Payee	Debit	Credit	Balance
101	1/01/01	Opening Balance		100.00	100.00
102	1/02/01	Steve Parker	10.00		90.00
103	1/15/01	The Rent	15.00		75.00
	1/15/01	Electric Co.	10.00		65.00
	1/20/01	Deposit		25.00	90.00
	1/30/01	Deposit		50.00	140.00

Figure 2. Positional balance display

There are, however, several problems with this code. First, in most cases, transaction files have multiple keys and the file browse use all (or several) of them. However, when changing tabs, calculations of the balance will be seriously impacted. The great secret of a positional balance is that only one sort order makes sense with this type of balance: date order. Displaying forward balances in number order or payee order or any other order is, simply, meaningless. (I have left a number order browse in the demo app to demonstrate the erroneous display that results when switching tabs.)

Second, when adding a record at the end of the keyed order, everything is fine. But if a new record is added so that it falls between existing records, the calculation will be erroneous. In Figure 3, the highlighted record was added after all the other records displayed. Note how the balance is "calculated." There is a definite problem here.

Number	Date	Payee	Debit	Credit	Balance
101	✓ 1/01/01	Opening Balance		100.00	190.00
		Steve Parker	10.00		90.00
102	1/15/01	The Rent	15.00		100.00
103	1/15/01	Electric Co.	10.00		115.00
110	1/15/01	Mortgage Co	50.00		50.00
	1/20/01	Deposit		25.00	75.00
	1/30/01	Deposit		50.00	125.00

Figure 3. Record inserted in the middle of the sort order

The problem shown in Figure 3 is caused by the way template-based browses read the disk file and create a view. The list box displays a queue, not the file or the view. This queue is created on the fly from the view.

So when a new record is added to the file, that record is also added to the view. Then it is added to the queue. The code in `SetQueueRecord` makes the required calculations and displays the list. Thus, the calculation uses the last value of `R_Balance`, the value in the last item in the queue before the add. It should be using the value in the item immediately before the new item.

I cannot get that value, at least not easily. However, because the queue contains only the number of elements that can be fit into the list box (typically around 15 for a browse like that shown above), it is not very expensive to completely rebuild the queue.

In `ResetFromAsk`, after `Parent` call, add this code:

```
R_Balance = 0
BRW1.ResetQueue(1)
```

and all is well again. The queue is rebuilt, starting with a zero balance (without resetting `R_Balance`, pressing the Enter key a few times produces ... interesting results).

The third problem is that if an item is added that causes the list to scroll down (or if the user scrolls to a new page of records), the balance will be off. This, too, is caused by the fact that the display queue contains only the records displayed and no history. So, the balance is recalculated from an intermediate value. Changing the browse from `Page Loaded` to `File Loaded` fixes this (if you use Quicken, now you know why it loads accounts so slowly).

At the bank, we had none of these problems. Of course, we did not have CRTs and produced only a printed report. A report template procedure is also much easier than a positional balance on a browse; you can find a sample procedure in the demo app.

Continually updated balance

There are several ways of recalculating an account balance, a "simple" balance calculating, if you will. My checkbook program still uses the legacy templates and most of the work is done in the form. The sample app accompanying this article is ABC and all of the work can be done in the browse.

In either case, the most important thing is to know *what* needs to be done. Where the calculations are executed is almost entirely a matter of taste; calculating in the browse or in the form will work.

The first thing to remember, in this scenario, is that the balance is stored in a control file, not the transaction file. So, (1) the appropriate record in the control file must be read early in the browse procedure and (2) the balance field in the control file must be updated periodically (being mildly paranoid about data integrity, after each item is added/changed/delete seems "periodic" enough to me).

In ABC browses, `ResetFromAsk` is called immediately after any file maintenance action, so this is a good place to maintain the balance. Before the Parent call all of the buffers are still intact, so this is where my code will go (in the original, legacy app, I used the update form's Ok Accepted embed; which you use is, as I said, a matter of preference).

Maintaining the balance is not a simple matter of incrementing the balance on an add and decrementing it on a delete. The debit/credit mark must be considered (here is one case where using the sign can be a modicum simpler). A new credit is added but a new debit is subtracted; a change in amount requires reversing out the previous amount (saved before calling the form) and putting "back" the "new" amount:

```
If Response = RequestCompleted !only if action completed
  Case Request
  Of InsertRecord
    If CHE:DRCR = 'D'           !debit
      ACC:BALANCE -= CHE:AMOUNT !update balance
    Else
      ACC:BALANCE += CHE:AMOUNT
    End
  Of ChangeRecord !remove previous amount
    !then add back "new" amount
    If CHE:DRCR = 'D'
      ACC:Balance += PreviousAmount
      ACC:Balance -= CHE:Amount
    Else
```

```

        ACC:Balance -= PreviousAmount
        ACC:Balance += CHE:Amount
    End
Of DeleteRecord
    If CHE:DrCr = 'D'
        ACC:Balance += CHE:Amount
    Else
        ACC:Balance += CHE:Amount
    End
End
End
End
Put(Accounts)           !update Accounts file
Balance_ = ACC:Balance !redisplay it

```

(Note that I write the recalculated balance with a "legacy" PUT statement, not an ABC method. Perhaps this is bad form, perhaps not, but it works perfectly well in this case.)

If changing the debit/credit mark is permitted (poor accounting practice, as is allowing changing an amount, which explains why users *never, ever* demand and developers *never* provide such a thing), the complexity of this code increases exponentially; I am not going to into that here.

Summary

Running balances are a rare case.

A positional balance is one of the rare cases where Page Loading is an absolute requirement. Simple balances require that changes be handled differently than adds or deletes. But once the business requirements are mastered, understanding how a Clarion browse behaves makes either sort of running balance fairly easy to implement.

[Download the source](#)

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

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 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

Steve Parker's 100th!

Published 2001-08-21

Okay, so Dr. Parker is no spring chicken, but he's not that old. The 100 refers not to his age, but his output of Clarion programming information. This week, the publication of (appropriately enough) [Running Totals](#) marks Steve's 100th Clarion article!

This is also Steve's ninth year in Clarion publications, beginning with the Clarion Tech Journal. Here's a complete list of Steve's articles, with links to those published in Clarion Online and Clarion Magazine.

<u>Abbreviation</u>	<u>Publication</u>
CTJ	Clarion Tech Journal
CWJ	Clarion for Windows Journal
COL	Clarion Online
CM	Clarion Magazine

1	CTJ	March/April 1993	A Still Better Validate Template For Pre-Release 3001
2	CTJ	May/June 1993	Clarion Database Developer's Dynamic Indices
3	CTJ	July/August 1993	Generating Expressions in 3.0
4	CTJ	September/October 1993	An Interview with Bruce Barrington
5	CTJ	September/October 1993	Let's Make a .Lib
6	CTJ	November/December 1993	Into the Real World with 3.0

[Search](#)[Home](#)[COL Archives](#)[Information](#)[Log In](#)[Membership/](#)[Subscriptions](#)[FAQ](#)[Privacy Policy](#)[Contact Us](#)[Downloads](#)[PDFs](#)[Freebies](#)[Open Source](#)[Site Index](#)[Call for](#)[Articles](#)[Reader](#)[Comments](#)

7	CTJ	November/December 1993	Opening and Closing Logos in 3.0
8	CTJ	January/February 1994	C3 Development Templates
9	CTJ	January/February 1994	The Double Loop Does ... You
10	CTJ	March/April 1994	Printer Control in 3.0
11	CTJ	March/April 1994	Beating Those Required Field Look Up Blues
12	CTJ	May/June 1994	An Alternative to the MultiPage Template
13	CTJ	May/June 1994	Ask Me Something Hard
14	CTJ	May/June 1994	.LIBs Agains
15	CTJ	July/August 1994	Multi-Paging Without the MultiPage
16	CTJ	September/October 1994	Q Basics
17	CTJ	November/December 1994	Using Browsers as Lookups
18	CTJ	November/December 1994	Updating Master Files with Append
19	CTJ	January/February 1995	Using Total Fields
20	CTJ	January/February 1995	Mastering CW File Loaded Drop Boxes
21	CTJ	January/February 1995	Reusability - It Never was Just for C++
22	CTJ	January/February 1995	Using 2.1's Environment() Function
23	CTJ	March/April 1995	Tools for CW
24	CTJ	March/April 1995	Distributed Processing with Clarion
25	CTJ	May/June 1995	Friendly Reports
26	CTJ	May/June 1995	Date Limits
27	CTJ	May/June 1995	How Forms are Called - CPD through CW

28	CTJ	July/August 1995	The Clarion Paradigm: How You Must Structure Clar. Prg.
29	CWJ	July/August 1996	MultiPaging in Clarion for Windows
30	CWJ	September/October 1996	Customizing Reports
31	CWJ	November/December 1996	Of Indices and Reports
32	CWJ	January/February 1997	Handcode Helper
33	CWJ	January/February 1997	A Case for Code
34	CWJ	January/February 1997	An xBase Survival Guide
35	CWJ	March/April 1997	Taking Control of Autonumbering
36	CWJ	March/April 1997	Time Stamping with Confidence
37	COL	July 1, 1997	Quick, it's CWIC
38	COL	September 1, 1997	Exploring the NAME attribute
39	COL	October 1, 1997	Queues: 1001 Uses
40	COL	December 1, 1997	CWIC Forever
41	COL	January 1, 1998	Marking Time, Part 1
42	COL	February 1, 1998	Marking Time: Round 2
43	COL	March 1, 1998	To Derive or not to Derive - That is the Question
44	COL	April 1, 1998	Calling Form Procedures
45	COL	April 1, 1998	Multiple Locators
46	COL	May 1, 1998	Exists()
47	COL	June 1, 1998	Using Indices in Reports
48	COL	July 1, 1998	Dynamically Generated HTML

49	COL	August 1, 1998	Lookups in C4
50	COL	August 1, 1998	Decaffeinating Forms and Browsers
51	COL	September 1, 1998	But it Doesn't <i>Look</i> Like IC!
52	COL	September 1, 1998	Pay the Money
53	COL	October 1, 1998	Images in Internet Connect
54	COL	November 1, 1998	Dual-Dual-Mode Apps
55	COL	December 1, 1998	Power broker...
56	COL	January 1, 1999	Dual-Hybrid Apps: The Details
57	COL	January 1, 1999	Filtering Browsers on the Fly
58	COL	February 1, 1999	The Best of Both Worlds: - Browse Boxes on the Web
59	CM	February 15, 1999	Don't Know, Do Care - A Philosopher Looks At OOP
60	COL	March 1, 1999	Update Forms without IC's Classes
61	CM	March 8, 1999	NAME() Comes Of Age
62	COL	April 1, 1999	Lookups: - You Don't Always Want to Validate
63	CM	April 5, 1999	How ABC Handles Multiple Sort Orders
64	COL	May 1, 1999	Order! Order! Order in the Files!
65	CM	May 10, 1999	How ABC Handles Multiple Sort Orders (Part II)
66	COL	June 1, 1999	Modifying IC's Basic page Formatting
67	COL	June 1, 1999	Recursive Lookups
68	CM	June 21, 1999	How ABC Handles Multiple Sort Orders (Part III)
69	CM	July 27, 1999	Working With Control Files I
70	CM	August 24, 1999	Alias - Who Was That Masked File?

71	CM	November 9, 1999	Clarion 5.5 Web Development Features
72	CM	November 16, 1999	Relation Trees: A Few Of The Finer Points
73	CM	January 11, 2000	WebBuilder Skeleton Basics: Which? When?
74	CM	January 18, 2000	WebBuilder Skeleton Basics II: Logos and Fonts
75	CM	January 25, 2000	Skeleton Basics III: Colors and Backgrounds
76	CM	February 8, 2000	Skeleton Basics IV: List Variations
77	CM	February 15, 2000	Skeleton Basics V: Listbox Relocation
78	CM	February 29, 2000	List Box Marking
79	CM	March 7, 2000	Advanced Skeletons: Managing Hyperlinks
80	CM	March 14, 2000	Skeleton Bidding And Selection
81	CM	June 14, 2000	The Clarion Advisor: Standard Address Handling
82	CM	July 5, 2000	Web Builder Reporting
83	CM	July 11, 2000	Sidebar Menus
84	CM	July 18, 2000	A Tale Of Three Brokers
85	CM	October 10, 2000	Template Writing Made Easier: The Template Wizard
86	CM	October 17, 2000	Web Development Options: An Overview - Part 1
87	CM	October 24, 2000	Web Development Options: An Overview - Part 2
88	CM	October 31, 2000	Give It a Nudge: Adjusting Report Position at Runtime
89	CM	November 7, 2000	Conditional Sort Orders and Page Breaks in Reports: Part 1
90	CM	November 14, 2000	Conditional Sort Orders and Page Breaks in Reports: Part 2
91	CM	December 12, 2000	Completely Dynamic Report Orders and Breaks Part 1

92	CM	December 22, 2000	Completely Dynamic Report Orders and Breaks Part 2
93	CM	March 27, 2001	Dynamic Filters: The Theory Behind the Facts
94	CM	April 3, 2001	Dynamic Filters: Applying The Theory
95	CM	April 10, 2001	SetFilter to the Max
96	CM	May 3, 2001	Replicating IDLE: All Quiet on the Keyboard?
97	CM	June 29, 2001	Handling Multiple Update Forms
98	CM	July 6, 2001	"Sometimes" Lookups
99	CM	August 16, 2001	Must Be in <i>This</i> List
100	CM	August 21, 2001	Running Totals

Reader Comments

[Add a comment](#)

**I'm not aware of any author with the duration and...
Way to go Steve! So, when does the movie come out?...**

Copyright © 1999-2001 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.

Find Expert Computer Consultants
Search by Skill or Location Worldwide FREE

published by
CoveComm Inc.

Clarion MAGAZINE

Migrating The Inventory Application To SQL Server (Part 2)

by **Ayo Ogundahunsi**

Published 2001-08-20

In [last week's installment](#) I gave an overview of converting a TPS application to SQL, and I began a conversion of the Inventory example application that comes with Clarion. I showed how to set table properties, create identity fields, install Jim Kane's auto-incrementing code, and create a connection string so the application can talk to the database. Now it's time to create the database and generate the table definitions.

SQL scripts

SQL Scripts are text statements that you execute on the backend to perform certain functions. These include creation of databases, tables, stored procedures; execution of scheduled tasks, and so many other operations that are normally done manually.

Clarion Enterprise Edition, version 5 and later, comes with a tool called the Synchronizer. This tool generates an SQL script from your Clarion dictionary, and stores that script in a text file. You then have to run this script on your backend in order to generate the database that matches your dictionary.

If you don't have Clarion Enterprise Edition, the alternative is to use third party, solutions most of which are free. Here are some programs you can use to generate SQL scripts:

- DumbDict (Developed by Tom Ruby). Can be downloaded at: <http://www.tomruby.com/dumbdict.zip>
- Geoff Bomford's Templates. Can be downloaded at: <http://www.comformark.com.au/gwbsql.zip>

DumbDict generates SQL Anywhere code, so the script needs some modification before you can run it on SQL Server. However, Dumbdict comes with the APP file so you can make modifications in the source. On the other hand, Geoff Bomford's templates generates pure SQL Server code as well as Pervasive.SQL code; this utility is indispensable if you do not have the Clarion

[Search](#)

[Home](#)

[COL Archives](#)

[Information](#)

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)



Synchronizer.

Generating scripts with Geoff's templates

After you've downloaded and installed Geoff's templates, open up the inventory application and choose Application|Template Utility from the main menu. Select GWBSQL-Step_3 (see Figure 6) to generate the SQL Script. Remember, you need to create a sub-directory called `scripts` under the directory that contains the template.

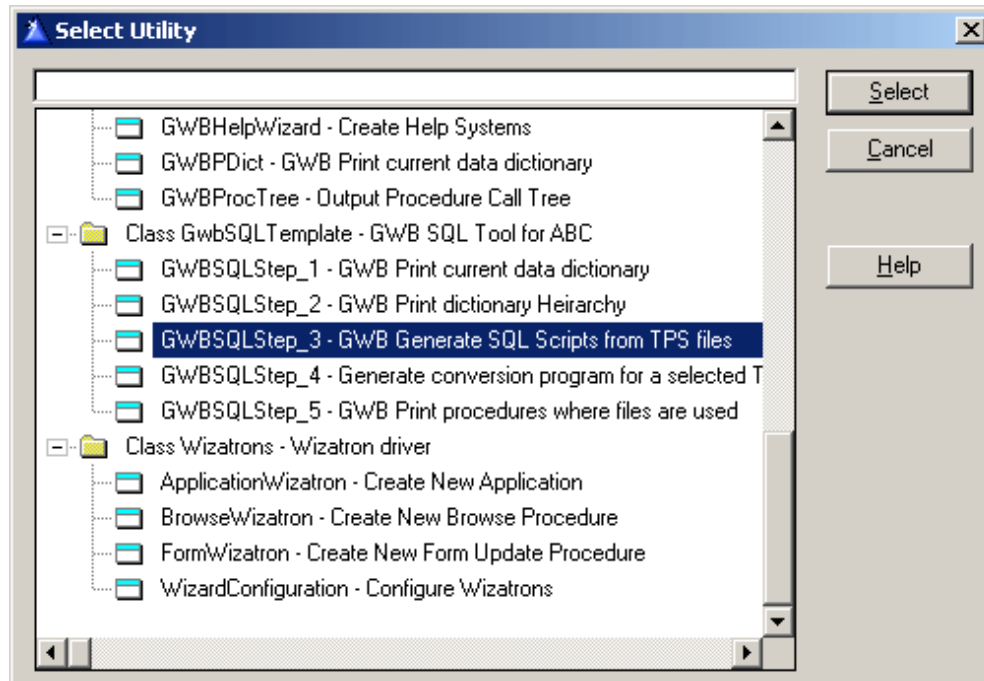


Figure 6. Generating scripts with the utility template

Generating scripts with the clarion synchronizer

Synchronization in Clarion Enterprise terms means ensuring that the structure of the Clarion dictionary matches the database structure at the backend. This implies that if you make a change at the backend, the synchronizer can modify the dictionary to match backend, or vice-versa (referred to as two-way synchronization). Though this is a desirable feature, it does not work well all the time, so what I recommend is one- way synchronization, i.e. from the backend to the Clarion dictionary. More about that later.

Creating a blank database

Before synchronizing the Clarion Dictionary, you have to create a blank database. You can do this by running a script or SQL Statement in the Query Analyzer, or by using SQL Server's command line utilities like OSQL.EXE, ISQL.EXE.

The command line utilities are the only way to run scripts if you are deploying Microsoft SQL Server Desktop Engine (MSDE). MSDE is a trimmed down version of SQL Server optimized for not more

than five users, and does not come with any of the tools mentioned under the paragraph "Tools of the Trade".

The ISQL utility is an old command line utility based on SQL Server's DB-Library API. This has been replaced by OSQL which is based on the ODBC API.

To create a database from the command line, enter the following command:

```
OSQL -SAYODAHUNSI\AYO_NV01 -Usa -Psa  
-Q"CREATE DATABASE INV_SQL"
```

OSQL will respond with messages like the following:

```
The CREATE DATABASE process is allocating  
0.63 MB on disk 'INV_SQL'.  
The CREATE DATABASE process is allocating  
0.49 MB on disk 'INV_SQL_log'.
```

To get a list of more options available using OSQL, type OSQL /?.

If you want to create the database using the Query Analyzer, the statement to run is: CREATE DATABASE INV_SQL, as shown in Figure 7.

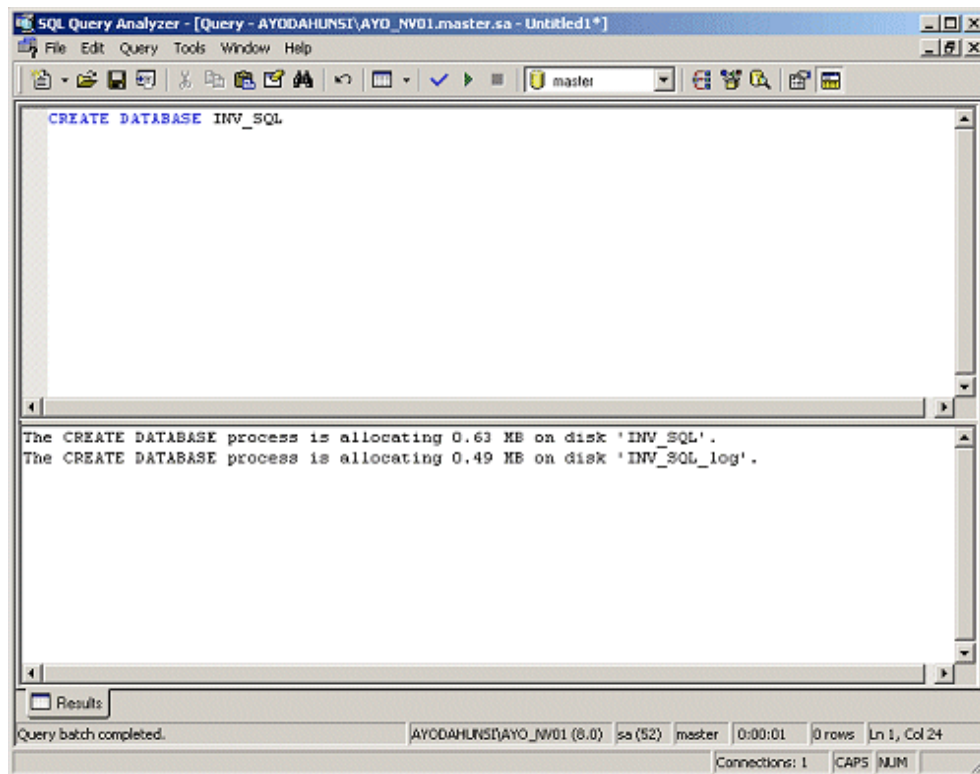


Figure 7. Creating the inventory database using Query Analyzer.

Different settings can be specified when creating a database for the first time. As you get more familiar with SQL Server, and you gain more experience, you can explore these options. For now, let

the default values take effect.

RI in Clarion dictionary fields

It is common for some people to attempt enforcing some level of Referential Integrity in fields using the Must be in Table option under the Validity Checks tab. The validity check settings in the VENDOR table is a classic example (see Figure 8). If you leave these settings unchanged, you will not be able to use the Clarion Synchronizer to create your SQL script. You must remove any Must be in Table requirements from your table field edits. This validity check will also cause a GPF when you run the Clarion Synchronizer

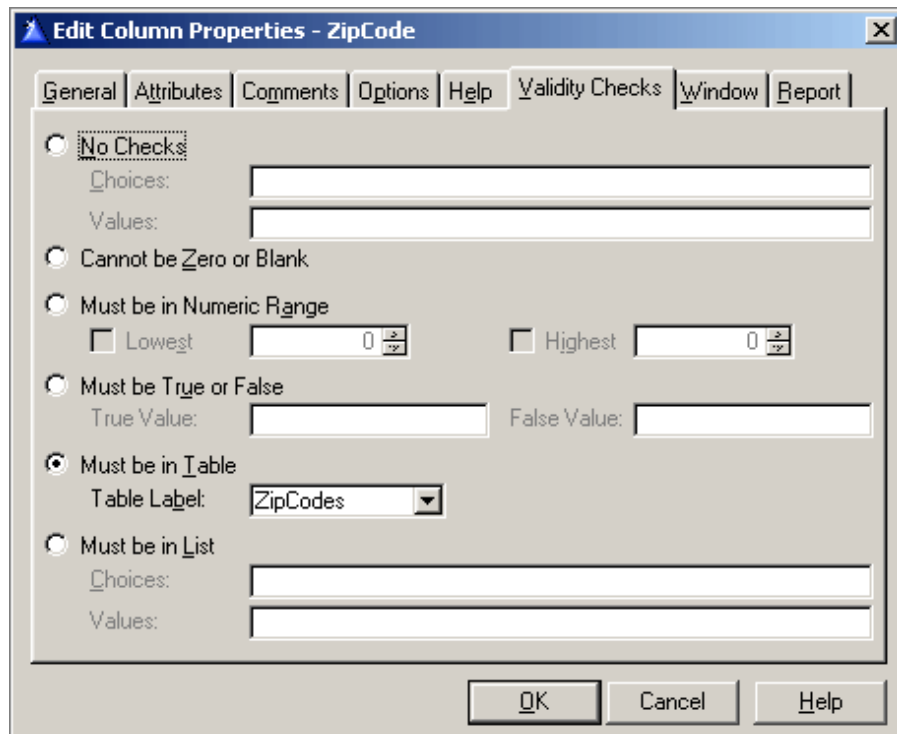


Figure 8. Clarion RI enforcement in the VENDOR table.

Default value in fields

You also have to be careful about how you set default values in your Clarion dictionary. For example, when you set defaults based on other fields, the Synchronizer will generate your SQL Script with these defaults, and SQL Server does not understand this. For example, the Default value for the Cost field in InvHist is PRO: COST, and the column is defined this way in the SQL script:

```
"Cost" DECIMAL(7,2), DEFAULT(PRO: COST),
```

Unfortunately, MSSQL does not know what PRO: COST means, so when you try to run a script created by the synchronizer you get a syntax error like this:

```
Line 46: Incorrect syntax near 'PRO:'.
```

When you use the Clarion Synchronizer to create your script, it is smart enough to use the equivalent SQL function for the Clarion `TODAY()` function set as a default in the `INVHIST` table. The function is `GETDATE()`, and the definition for the `Date` column becomes this:

```
"Date" INT NOT NULL DEFAULT (getdate()),
```

But there is another caveat here. You can see that the data type used is `INT`, which is equivalent to `LONG` in Clarion! I'll cover this in more detail later. A reminder: do not use `LONG` for date types; use `DATE` instead.

Script generation

Assuming you've followed all suggested changes, you can now run the synchronizer and have it generate a script.

Tools of the trade

There are a couple of tools that come with SQL Server that have had a great impact in the way people work with SQL Databases. These tools make it quite easy maintain, test, and even deploy applications using SQL Server while the developer is still in the process of acquiring SQL skills. Though there are a lot of books available that describe how to effectively use of these tools, you can quickly find your way because of the intuitive interfaces.

- Enterprise Manager (EM) – As the name indicates, this tool manages everything from databases to stored procedures, Alerts, Jobs, replication, etc.
- Query Analyzer (QA) – The Query Analyzer is used to run scripts or SQL commands. Commands can also be parsed (i.e. checked for accuracy) before they are executed. Press F5 to run either the highlighted query, or every command in the QA window. Press Ctrl-F5 key to parse the highlighted query, or every command in the QA window. Almost all the tasks performed by pointing and clicking within the Enterprise Manager can be done within the Query Analyzer. The developer new to SQL Server should stick to the Enterprise Manager in order to get familiar with SQL.
- View Designer (VD) – The View Designer is seldom known to new users of SQL Server, and as a result these users may spend too much time figuring out how a complex `SELECT` statement should be written. When I am creating `SELECT` statements whereby I join many tables, I usually use the View Designer to graphically build up the kind of join I want. Now, if your identity fields were defined as I suggested earlier, once you populate the View Designer the correct links/joins to your database are automatically created and scripted.
- Database Designer – The database designer shows you how tables are related/linked in your database. This tool can also be used in creating new tables, and modifying the properties

of existing ones. Note that the relations in the tables (as shown in Figure 9) are not generated automatically when you synchronize with a Clarion Dictionary; you will have to add these yourself.

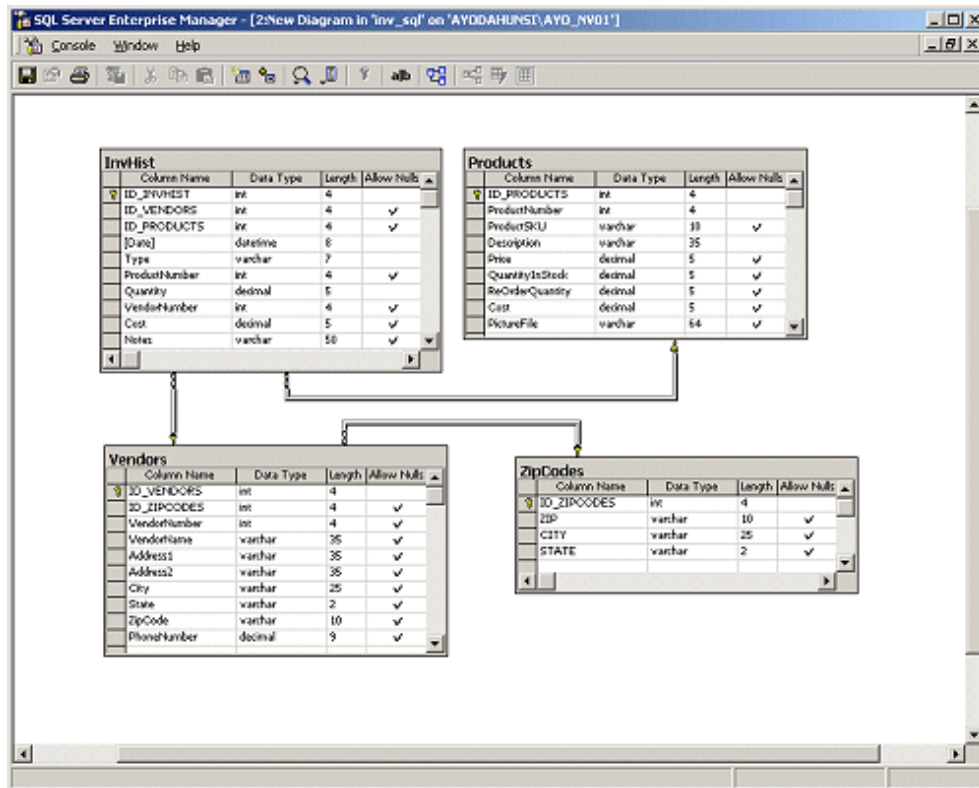


Figure 9. The Database Designer.

However, if you followed the suggestions in creating ID fields and their corresponding keys as described under Step 1 – Table Changes, the links are generated automatically based on fields with similar names.

- DTS (Data Transformation Services) Designer – This is probably one of the most effective tools in SQL Server for data conversion, and beats anything I've seen on other database backends. The DTS Designer can be used for converting data from one file format to another with a high degree of automation and sophistication. As an example, think of a scenario where you have to download customer invoices from an AS/400 mini computer with some massaging of the data required before it can be used; you can do this with DTS. Or perhaps you want to do a parallel run between an old system and a new one, and data is still being entered through the old system; you can set up DTS to synchronize the two systems. In a future article, I will show how to transform the data (in the original Inventory Example) to this SQL Server based Inventory system.
- Books Online (BOL) – This is the SQL Server help system. It is quite versatile and indispensable, especially for a starter.

That's all for this week. [Next week](#) I'll look at some workarounds for Clarion date handling problems, and I'll show how to run the

table creation script. I'll also complete the basic application conversion.

[Download the source](#)

[Ayo Ogundahunsi](#) presently lives in Henderson, Nevada, about ten minutes from Las Vegas. He works for [Impac Medical Systems Inc.](#), the leading company in cancer therapy software (written in Clarion). Impac has its headquarters in Mountain View, California. Ayo is married to Ayodola, and they have two boys, Darren and Joshua.

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 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

Creating Filter Expressions The Easy Way

by John Morter

Published 2001-08-16

Creating filter expressions can be a tricky process - there's no "Populate Column" toolbox readily available and it's possible to make a typing error that's not picked until run-time (when you get the dreaded "bind" error).

Here's a nifty workaround that can save some grief. For each variable-name you need, go down to the AdditionalSortFields entry field and click the [...] lookup button. This will bring up the File Schematic dialog, from which you can select the fields you need.

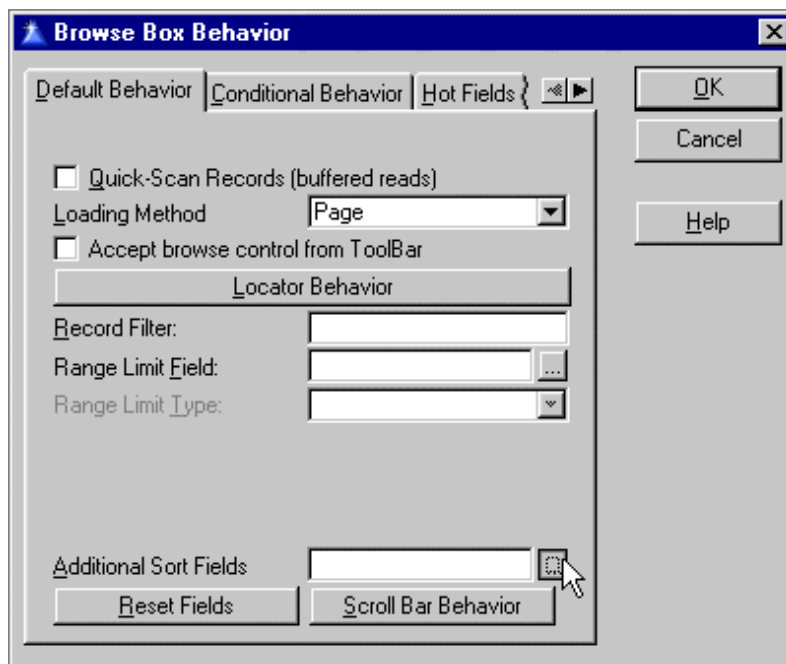


Figure 1. Using the Additional Sort Fields lookup

Cut and paste the selected fields into your Record Filter expression, and just add whatever comparative operators the filter requires. Remember you can also select (and create) local and global variables from the File Schematic dialog.

[Search](#)

[Home](#)

[COL Archives](#)

[Information](#)

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)



[John Morter](#) is a member of the Victorian Clarion Users Group (Melbourne, Australia). His moneymaking day job doesn't actually involve Clarion (at least not officially), but Clarion occupies a lot of his spare time as a hobby to keep his tech-developer background up to date. He sails in the bay during the summer on his racing catamaran named Flat Chat, which is Australian slang for "at top speed" - or "at high velocity".

Reader Comments

[Add a comment](#)

Neat trick.

Can also use the Formula editor to build a filter...

I use cutting and pasting from the "Hot" tab since I have...

Copyright © 1999-2001 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

Must Be in *This* List

by **Steven Parker**

Published 2001-08-16

"Must Be in List" validation does little more than complete template prompts to call a lookup (as in Figure 1). What if you need to validate against a specific list? I am writing an app for the Native American Foster Parents Association. When a child is entered, the parents are also entered (yes, a *real* parent-child relation). For various reasons, the child's tribal association, if known, is needed.

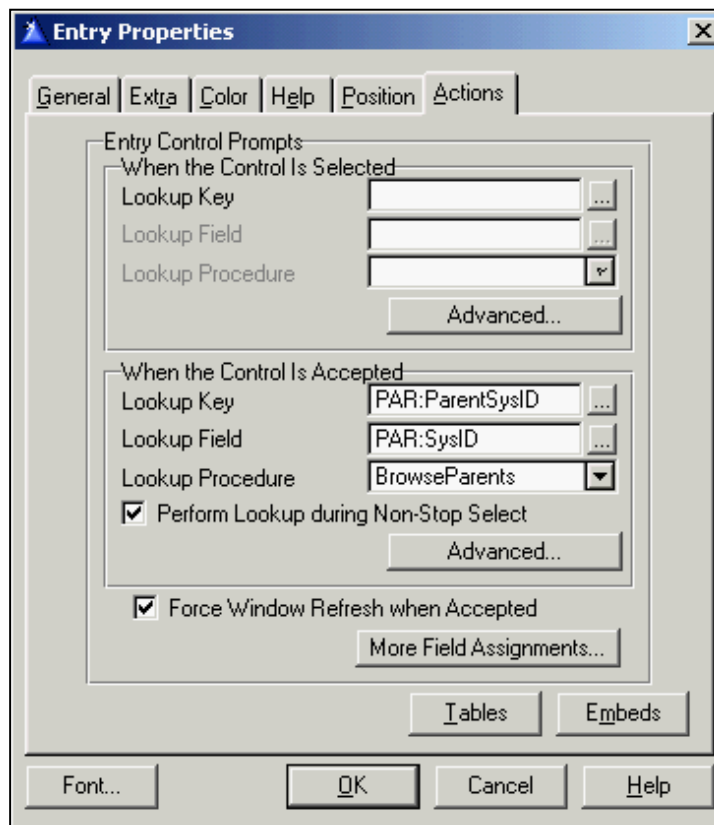


Figure 1. "Must Be in List" Validation

Now, and this is the interesting part, the child's tribe *must* be one of the parent's tribes (a Menominee and an Ojibwa can't have a Navajo child).

[Search](#)

[Home](#)

[COL Archives](#)

Information

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

Downloads

[PDFs](#)

[Freebies](#)

[Open Source](#)

Site Index

Call for

Articles

Reader

Comments



I could do a lookup into the Tribes file and compare the results to the mother's tribe (if entered) and to the father's tribe (if entered). If I fail to get a match, I could present a message.

This will work but strikes me as very sloppy *and* very unprofessional looking. When entering or updating a child's record, I've obviously got to look up the mother and father records to get the information to check. Why can't I customize the tribe list that the user selects from information I've already got?

Why not indeed. Why would I present the entire list of 70+ tribes when no more than two are eligible?

I could set up the tribe lookup window so that it filtered on the parents' tribes. But not only does this seem like an awful lot of work, a lookup window to display not more than two records is just plain wasteful. A file drop would be perfect for the purpose.

By populating a list box without the template (i.e., control only), I can name the queue used to fill it (I am not constrained to using an existing file as I am with the File Loaded Drop control template) and how many queue entries to show (see Figure 2).

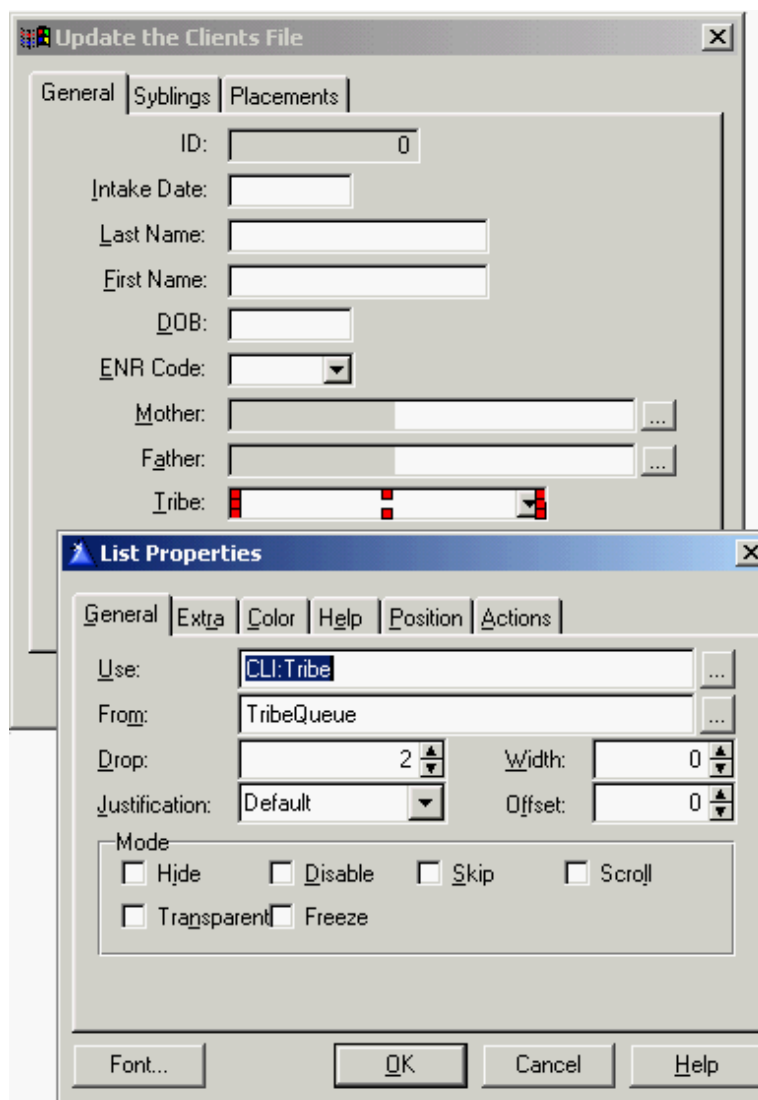


Figure 2. List control masquerading as a File Drop

The TribeQueue is declared locally, using the Data button as in Figure 3.

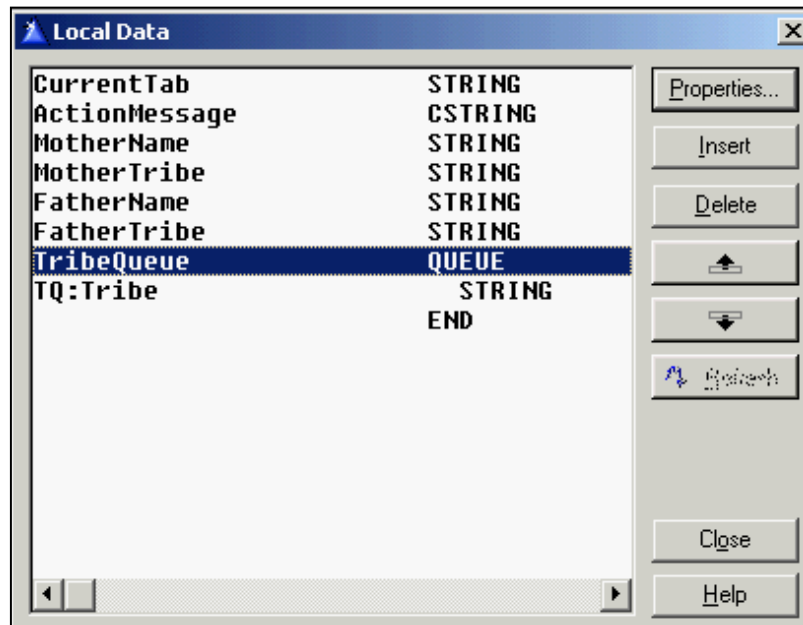


Figure 3. Locally declared queue

All that remains is to ensure that the queue has the information needed. Since I am looking up the parents anyway, it is just a matter of five simple lines of code to ensure that the parent's data gets into the local queue (two lines) without duplication (three lines):

```

PAR:SysID = CLI:MotherID
If ~Access:Parents.Fetch(PAR:ParentSysID)
  MotherName = Clip(PAR:FirstName) |
    & ' ' & PAR:LastName
  MotherTribe = PAR:Tribe !prime local variable
  TQ:Tribe = MotherTribe
  Get(TribeQueue,TQ:Tribe) !check for value in queue
  If ErrorCode() !if not there
    Add(TribeQueue) !add it
  End
End

```

To handle all possible circumstances, inserting records and changing them, I call this code (note that it is called a second time for the father) in INIT and after completing each of the parent fields. A local method or a procedure routine is ideal and, to prevent "premature selection" on an add, I disable the control until at least one parent record has been selected, as in Figure 4.

The screenshot shows a software window titled "bill smith" with a close button in the top right corner. The window has three tabs: "General", "Syblings", and "Placements". The "General" tab is active and contains the following fields:

- ID: 1
- Intake Date: 8/03/01
- Last Name: Redsmith
- First Name: Bill
- DOB: (empty)
- ENR Code: (dropdown menu)
- Mother: Anne Able (with a browse button "...")
- Father: Bob Baker (with a browse button "...")
- Tribe: Navajo (dropdown menu)
- DCFS Case ID: Tribe (dropdown menu)
- DCFS Date: Navajo (dropdown menu)

At the bottom of the window are "Ok" and "Cancel" buttons.

Figure 4: The final entry form

When an entry must in a list but not just any list, create your own.

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

Reader Comments

[Add a comment](#)

This makes the interface 'sing'. Excellent and clear. Thank...

Copyright © 1999-2001 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

True Threading: What You'll Need To Know

by **Dave Harms**

Published 2001-08-15

Multitasking is an important part of Windows application development; we've all become used to running multiple applications at once, and most of us create MDI applications which can have multiple windows open, each on its own thread. Clarion does this sort of thing well, for the most part. So what's the problem, and why are developers pushing for "true" threading support in Clarion?

Clarion's support for multitasking falls loosely into the "cooperative" category. In cooperative multitasking each thread must yield control to other threads, or the other threads cannot run. The Windows operating systems (at last since 3.x) support varying degrees of preemptive multitasking, where the OS assigns tiny slices of processor time to each thread in turn. When most developers talk about "true" thread support, they're referring to preemptive OS threads. Such threads are ideal for background and time-critical processes, and would be a welcome addition to Clarion.

But the change from cooperative multitasking to preemptive multitasking isn't without its difficulties. In this article I'll attempt to explain the differences in the two models, the problems with Clarion's current implementation, and suggest what Clarion developers can expect if and when preemptive multitasking is directly supported in a future version of Clarion.

I've always had unanswered questions about how Clarion's threading works, so I consulted Andy Ireland and Jim Kane, two developers who have a lot of experience working with, and around, Clarion's threading model. The technical goodies are theirs; the errors of fact mine.

Understanding threading

In Windows, any running application is a process, and any process can have multiple threads of execution. In a way, having multiple threads in a process isn't that different from having two processes running at once on the operating system; the difference is

[Search](#)

[Home](#)

[COL Archives](#)

[Information](#)

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)



primarily one of scope, since the threads can share data in the application (process) which originally spawned them.

When you have multiple threads of code executing, you either need multiple processors (one thread per processor) or you need a way of switching between threads so each gets its share of processor time. In reality most multi-processor systems need a way of switching between threads anyway, as there will likely be situations when there are more threads running than there are processors available.

In preemptive multitasking, the OS interrupts whatever one thread is doing so that another thread can continue to execute. The state of the currently running thread is saved, the state of the thread to resume is restored, and then that thread gets a few clock cycles, until it's time for the next thread to briefly execute. The individual slices of processor time are so small that threads appear to execute simultaneously.

Clarion's threading

In Clarion (as we now know it) you begin a thread with the `START` function. The Help for `START` states:

Code execution in the launching thread immediately continues with the next statement following the `START` and continues until an `ACCEPT` statement executes. Once the launching thread executes `ACCEPT`, the launched procedure begins executing its code in its new thread, retaining control until it executes an `ACCEPT`.

Note carefully what's happening with `START`. When, for instance, you choose a browse procedure from a main menu (where the browse is `STARTED` on its own thread), the code in the main menu procedure (usually an application frame) *continues to execute* until it reaches its own `ACCEPT` loop. The `STARTED` procedure does not begin executing right away, because Clarion's thread switching code is contained inside the `ACCEPT` statement. Once the main menu's `ACCEPT` is called, the `STARTED` procedure begins to execute, and it runs, exclusively, until its own `ACCEPT` is called (or, in a 16 bit application, `YIELD` is called).

This is an example of cooperative, not preemptive, multitasking, since execution doesn't switch from one thread to the next until `ACCEPT` says it can. However, Clarion's `START` function does call the `CreateThread` API function, so a Clarion thread really is a true preemptively multitasked operating system thread under the hood. It just isn't allowed to run as one because Clarion's runtime library (RTL) is not designed to be used with multiple preemptively switched threads.

The problem with threads

The real problem with preemptive multitasking is that there's no

way for the developer to predict when two threads will attempt an operation on the same data. Clarion string functions are a prime example. In most cases, when you call a Clarion string processing function, the string you're operating on is pushed onto the Clarion string stack, which is an area of memory shared by the application's threads. (If in debugging a section of code you see the `Clas$Pushxxxx` and `Clas$Popxxxx` functions, you know the operation is not safe.) If no other thread attempts a string operation before the first completes, all is well. But if the threads are preemptively multitasked, then a second thread may begin a string operation before the first thread's string operation completes. The result will be bad data at best, and a GPF at worst.

Code which can tolerate unpredictable thread switching is called *thread safe*. Unfortunately, the Clarion RTL is, at present, not thread safe, so it has to resort to other methods to ensure that there will be no dangerous overlapping calls to library functions. As you've no doubt guessed, that method is to mediate all thread switching with the `ACCEPT` statement.

NOTE: Although the RTL isn't *thread safe*, don't misinterpret this to mean that the RTL isn't *safe*. It is. It's just designed for a cooperative threading model.

ACCEPTing thread changes

When you launch a number of threads in a Clarion application, each of those threads is a preemptively switched operating system thread. But the Clarion RTL only allows one of those threads to be active at any one time. All the rest of the threads are in a wait state, so when the OS assigns processor time to one of those inactive threads, that thread immediately returns control to the OS. Although the OS is continuously slicing up processor time for each thread, only one thread in a Clarion application is actually doing anything at any one time. And that means that two threads will never call the same RTL function at the same time, guaranteeing no corruption of data.

True threading in Clarion

It is possible to have true preemptively timeshared threads in Clarion if you use the Windows API. See Jim Kane's [article on threading](#) for details, or have a look at the Winsock support classes in `ABAPI.INC/ABAPI.CLW` in your `LIBSRC` directory.

Any threads you create with the API won't be managed by the Clarion RTL, so they will not require, or be affected, by `ACCEPT` statements. The corollary is that you must be very careful about what kind of code you write for that thread! You should stick to data types like `CSTRING` and `LONG`, and use only thread safe API calls to operate on data. Limit your use of strings for buffers, and use the various Windows memory allocation functions to ensure your code is not working with shared data.

Getting out the crystal ball

If preemptive threads seem like too much hassle right now, you can wait for a future release of Clarion with better support. SoftVelocity has indicated that work is under way on replacing the current threading model with "a more OS standard method." No timeframe has been announced, and while it's not clear that this means a completely thread safe RTL, I'll go out on a limb and say that I expect to see just that no later than Clarion 6. I trust I'll be reminded by a reader or two if this turns out to be a bad prediction.

A version of Clarion that supports full preemptive threading will probably generate some slightly different code than is now the case. Here are a few thoughts on what changes to expect.

Global data

Both Legacy and ABC applications rely on significant amounts of global data for files/tables. Even if you have five threads operating on one table, and the table is threaded, there is really only one active buffer for the table. Each time you switch threads, the `ACCEPT` statement triggers a context switch, saving the buffer's current contents and restoring the contents for the thread you've switched to. This clearly wouldn't work in an application where multiple threads of execution really can execute concurrently. You need to have separate active table buffers for each thread. The same goes for the various global objects such as `FileManager` and `RelationManager` instances. Since thread switching will no longer be under the control of the `ACCEPT` statement, separate instances of these objects will have to be explicitly created for each thread.

Similarly, it might no longer be wise to use global variables (`GlobalRequest`, `GlobalResponse`) to communicate between browses and forms. Remember that although these variables have the `THREAD` attribute, it's the `ACCEPT` statement that mediates their allocation, and that will no longer be appropriate. All of the memory management currently done by `ACCEPT` will have to be handled elsewhere, and in a thread safe manner.

Managing shared access

Some data sharing situations can be solved by giving each thread its own copy of the data, but other times threads really do need to share data. Since it's never safe for two threads to operate on the same data at the same time, you need a way of serializing access to that data. In Windows API programming, this is called creating a critical section. Jim Kane gives the following example (abridged) in [Part 2](#) of his API threading article:

```
EnterCriticalSection(address(critical_section)).  
...  
LeaveCriticalSection(address(critical_section)).
```

The `EnterCriticalSection` API call takes as a parameter a group structure which the API call uses to tell any other threads that this section of code is now in use by another thread, and can't be executed until the `LeaveCriticalSection` function is called. I would expect a future version of Clarion to have a set of simple wrappers for these functions.

When you mark a block of code as a critical section, you have to keep in mind that you're effectively creating a bottleneck. Since only one thread can execute that code at a time, if the synchronized code takes a long time, you may end up with a number of threads whose execution is seriously delayed. It pays to make your critical section as small and fast as possible. Remember, use critical sections only when absolutely necessary! If you can work around some or all of the code with local non-shared data, do so.

Daemons and other idle threads

Sometimes you want a process to run in the background, but most of the time it won't actually be doing anything. It's just waiting for orders. Perhaps you want it to check for email every sixty seconds or so. Currently, you have to do this with a `TIMER` attribute on a window. A true background process (often called a daemon) won't need a window or a timer; instead I would expect to see a `wait()` or `sleep()` function which you could execute inside a loop. See the `WaitForSingleObject` API call for an example.

I think daemons are one of the most intriguing aspects of true threading. They're great for everything from print queues to reminders to background email checking or other communication tasks. They're particularly useful for web development, where servers sit idle much of the time, waiting for requests for information.

Summary

For the most part Clarion applications are not so much multithreaded as serially single threaded. The blocks of code that execute between thread changes are generally quite large, compared to the small slices of processor time in preemptive multitasking system. The upcoming move to true threading means Clarion developers will gain new functionality, but will also need to become aware of thread safety issues, and in particular manage access to shared data in a way that not only prevents data corruption, but allows threads to run with a minimum of interruption. True threading will also make it possible to create daemon processes, which offer tremendous utility to the developer.

*[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](#)

**I have a situation in a 32 bit program where it appears...
One place where the current Clarion threading falls down
is...**

Bruce, Andy I might be able to clear this up - he did...

Ok, the confusion is in how yield is implemented so I did...

I thought I should clarify something further... In the...

**What I've often wondered, is if you are creating a COM
sink...**

It's not true Andy... [from the docs] >Within your...

Very interesting article Dave! For those who would like...

I agree with you Bruce but my last comment was merely...

COM Sink synchronisation ideas Hi Jim, What you are...

Andy, Thanks for responding. I like the sendmessage()...

NCT functions No, the Nct variants mean 'Non clarion...

Copyright © 1999-2001 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

Migrating The Inventory Application To SQL Server (Part 1)

by Ayo Ogundahunsi

Published 2001-08-15

The amount of data being processed in corporate databases and over the Internet has created a demand for more powerful engines for data storage, access, and processing. While Clarion is a powerful RAD tool, a necessary complement is an excellent backend SQL Database.

In this article (the first in a series) I will attempt to reinforce information contained in earlier articles (see links below) dealing with conversion from a flat-file system to an SQL system. The emphasis of this series is on portability, business rules, referential integrity (RI), and Clarion as an interface tool. More specifically:

- If you are the designer of the overall application, i.e. you are responsible for creating the original tables, the business rules and logic of the application must be moved to the backend so that a non-Clarion application will be able to use your logic. I call this approach *non-isolationist* in the sense that components, for instance your data types, will be directly accessible to other systems. For example, you do not use a LONG instead of a DATE data type so that updates from within Clarion will not differ from updates outside of Clarion.
- If your application will be accessing an existing database, then you will still try to put most of your business logic/rules at the backend, i.e. in stored procedures.
- When RI is enforced at the backend, you greatly minimize the risk of bad data entering into your database. This becomes more evident where other applications input data to be used by your system.
- Clarion moves away from its role as the complete database application and into a role as an interface tool. Extensive data manipulation is done at the server using stored procedures called from within Clarion.

Some may argue that the points mentioned above reduce control you have over the customer, especially in the case of contractors. Nevertheless, it is a matter of how you view the services you are offering. Do you want to deploy a system where all the

[Search](#)[Home](#)[COL Archives](#)[Information](#)[Log In](#)[Membership/](#)[Subscriptions](#)[FAQ](#)[Privacy Policy](#)[Contact Us](#)[Downloads](#)[PDFs](#)[Freebies](#)[Open Source](#)[Site Index](#)[Call for](#)[Articles](#)[Reader](#)[Comments](#)

components can *only* run with Clarion? Or, you want to deploy a system for which someone can easily code a Visual Basic interface? It is a business decision. Note however that times are changing, and customers are starting to look at more portable systems, so you may not want to be overly proprietary in your design.

On the marketing side, the more you emphasize on the fact that the logic of your application resides in SQL, the easier it becomes for you to enter into the arena controlled by other languages like Visual Basic, PowerBuilder, etc.

This series of articles specifically targets MSSQL Server, and will present a step-by-step approach to an existing application. In later articles I'll continue to build upon the converted application with enhancements that will allow it to mature. The application will be subjected to modifications which simulate client requests for changes due to flexibility, scalability, etc. If you wish, you can post your own requirements (using the comments feature at the end of this page), just as clients request changes in real life situations.

Existing resources

There are many articles in ClarionMag and elsewhere that have treated similar conversions extensively; please make sure you refer to them since I'll assume some prior knowledge. Here are some (free) articles you can read:

- [How To Convert Your Database To SQL](#) by Scott Ferret
- [Stephen Mull's Guide To Converting To MS-SQL](#)
- [MS-SQL Tips and Tricks and C5](#) by Rick Hoffman

The second article (a must read) covers a lot about conversion and contains most of the tips and tricks explained in the third article.

See the end of this article for some non-Clarion resources.

Getting Started

There is an Inventory example that comes with Clarion. It is located in:

```
C:\Clarion5\Examples\INVNTORY
```

for Clarion 5, or in

```
C:\C55\Examples\INVNTORY
```

for Clarion55. To follow along with this article, make a copy of this directory to C:\INVNTORY. All the examples will be based on this directory.

I will be using the terms Columns and Fields interchangeably, also Rows and records. Columns and Rows are SQL terms; Fields and

Records are the Clarion equivalents.

The conversion steps on the Clarion side are as follows:

1. Dictionary /Application Changes
 - a. Create a copy of `INVNTORY.DCT`, `INVNTORY.APP`.
 - b. Change the Table driver, properties.
 - c. Add Identity fields.
 - d. Change Data types (if needed. E.g. `LONG` to `DATE`).
 - e. Remove Initial Values from Clarion dictionary. If `TODAY()` is used and is needed, remember to define a `DEFAULT` in SQL Server as explained under the `DEFAULT` section in this article.
 - f. Remove `GROUPS` if used (in SQL tables you will only use `GROUPS` to translate the SQL `DATE/DATETIME/TIMESTAMP` data types to their Clarion equivalents).
 - g. You will also need to change `STRINGS` and `MEMOS` to `CSTRINGS`. Remember to add 1 to the size of your field when using `CSTRINGS`. Do not use the `LONG` data type for date fields; use `DATE` instead.
 - h. Delete procedures in `APP` file.
2. Template/Classes
 - a. Auto Incrementing – The Jim Kane Solution
3. Connections
 - a. Connection String
4. Database Creation
 - a. RI in Clarion Dictionary Fields
 - b. Default values in Fields
 - c. Script generation (Synchronizer)

Step 1 - Dictionary /application changes

You have to load the Inventory dictionary located in `C:\ INVNTORY` and save it as a new dictionary under the name `INV_SQL.DCT`. Repeat this step for the application file: `INVNTORY.APP` saving it as `INV_SQL.APP`. One thing you shouldn't forget to do is to change the dictionary in `INV_SQL.APP` to `INV_SQL.DCT`.

The next thing is to modify the table properties by changing the Driver to MSSQL Server, Specifying an Owner, and the way the table is named on the backend. Figure 1 shows what the `InvHist` Table looks like after this step is complete.

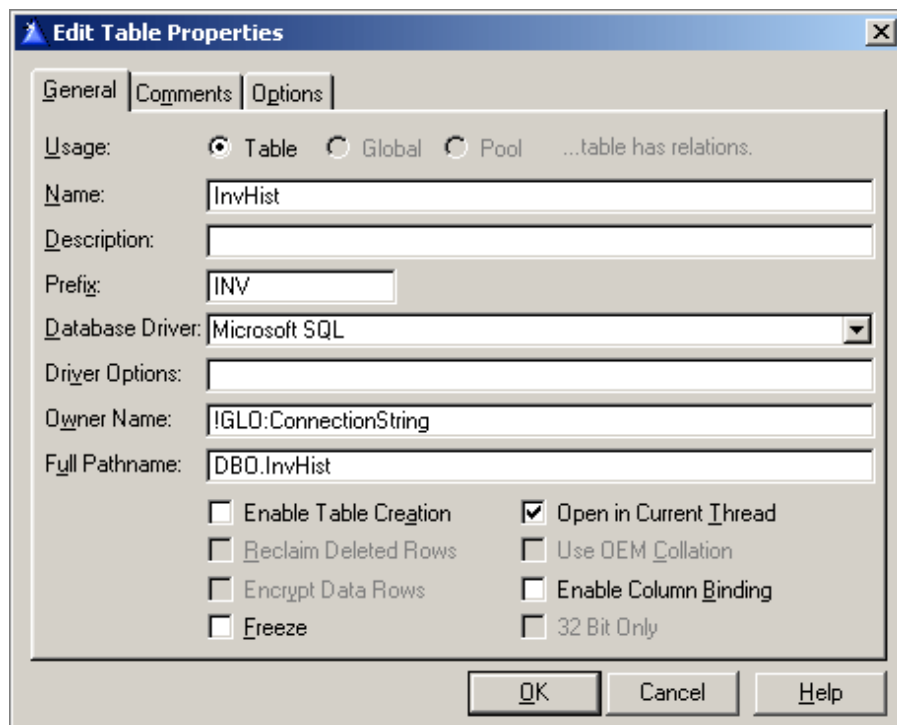


Figure 1. Changing the InvHist table settings

The fields that have changed are the driver, the owner, the full pathname, and the Enable Table Creation checkbox.

Owner Name: (!GLO:ConnectionString)

The owner name is the label of the connection string (see explanation on Connection String below). You will have to define this as a Global variable either in the Dictionary or in the application. The ! prefix tells Clarion this is a field and not a string literal.

Full Pathname: DBO.InvHist

The MS SQL backend recognizes the pathname like this. DBO stands for Database Owner, and InvHist is the name of the table. For now, I will stick to this simple approach to ownership. A complete understanding database ownership and roles is beyond the scope of this article.

Enable Table Creation: Unchecked

Enable table creation is unchecked deliberately. It is better you create the tables with the backend than through Clarion.

System IDs and identity columns

Properly designed tables usually have fields (columns) designated for storing the identity values of rows. The identity value is unique for each row in a table, and can be used to link child records in other tables. This subject has been already covered in detail by Dave Harms in his [SQL articles](#).

If you load the Inventory dictionary into the Data Modeller, you can see that there are no identity fields defined. **Products** does have a **ProductNumber**, and **Vendors** has a **VendorNumber**, which are presumably unique in each table, but these data could also change. It's better to have a unique, autoincremented number which will never need to be changed.

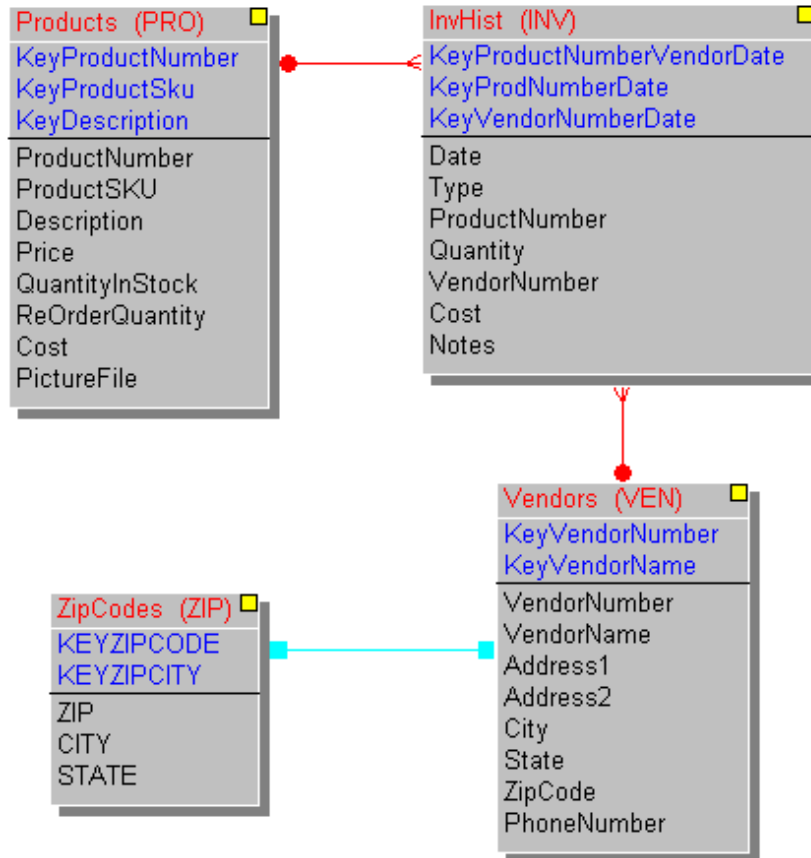


Figure 2. The Inventory tables without identity fields

After adding the identity fields, I have something like this:

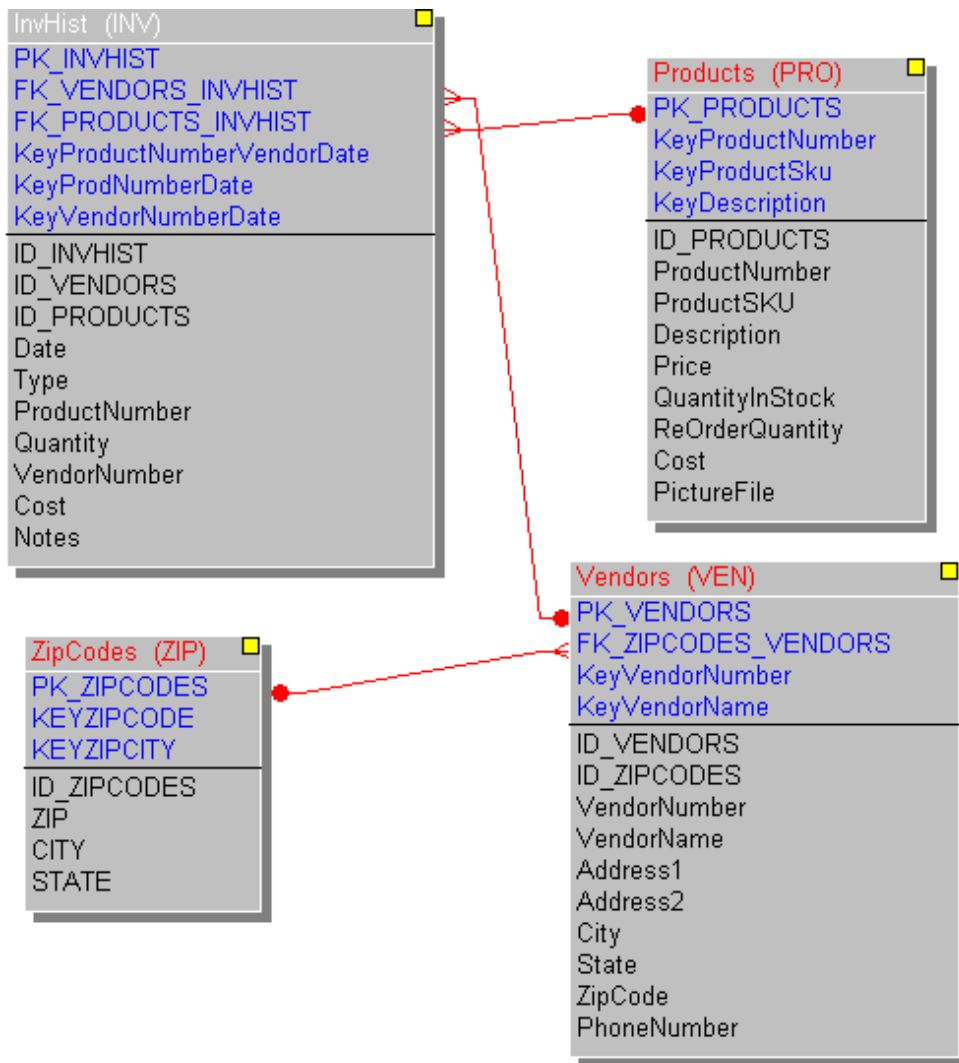


Figure 3. The Inventory fields with identity fields added

A couple of things to note here about naming and key attributes. The naming convention is up to you. Some people use `SysID`, some `NameOfTable_ID`. I prefer using `ID_NameOfTable` for the simple fact that in a very big database, you can immediately see the linked tables, sort the IDs and put them together in a big table, etc. Also, if you are using the MSSQL Server View Designer (more about this later), once you populate the screen with tables, the designer automatically connects the tables together for you. This is quite helpful and makes your work a lot easier. I use Crystal Reports to print out my database structure, and I have the fieldnames sorted, so I have all the link fields (those starting with `ID_`) all together.

I used the prefix "PK" to indicate Primary keys, and "FK" to indicate Foreign Keys. It is always better to use a naming convention like this because when you synchronize with the MSSQL Server, you will find the Table design interface in the Enterprise manager easier to use.

I can also go further and rename the other keys like `SK_VendorNumber` where "SK" means sort key, or `UQ_VendorNumber` where "UQ" means unique key. It is good to

distinguish these keys in a way you can easily recognize because it quickly gives you an idea about how the key was created.

In setting the attributes of the keys make sure that all your keys are case-sensitive. If you don't do this, performance will probably suffer significantly.

Deleting the application procedures

All procedures in the application file `INV_SQL.APP` except the `MainFrame` procedure are to be deleted. Yes, deleted. Once you are done with initial stages of conversion, you can import the same procedures from the original application file (`INVNTORY.APP`). This makes conversion faster.

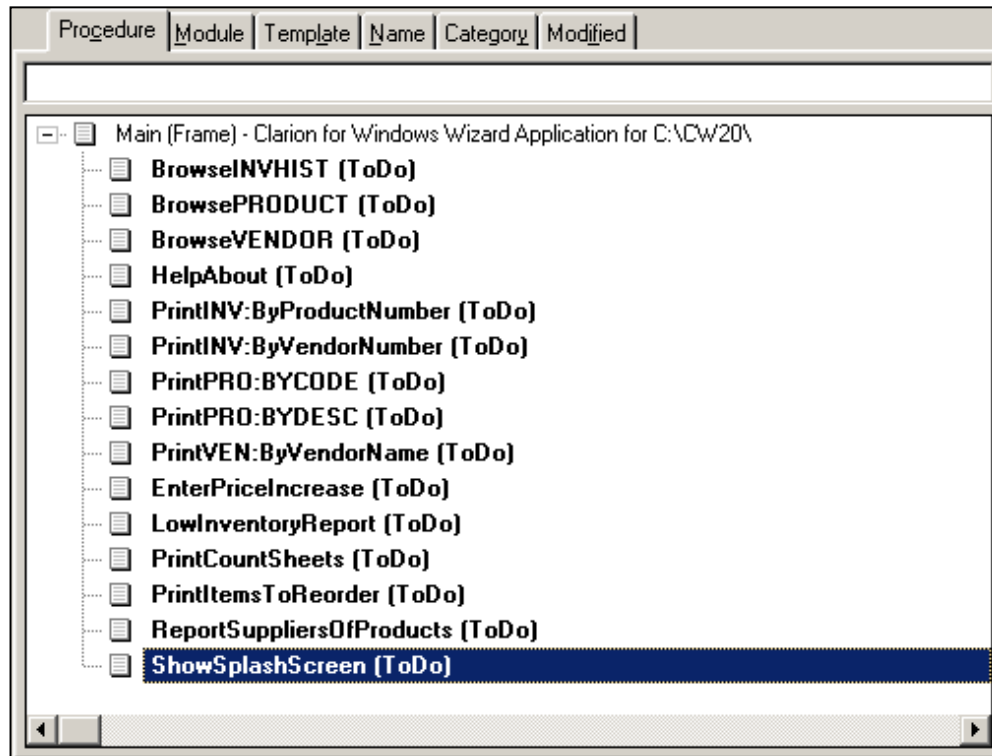


Figure 4. Deleted procedures

Auto Incrementing

In short, the problem with SQL auto-incrementing is that while it's safe and efficient to have the server create the auto-incremented ID for a new record, you won't have that ID available to you when you want to add child records to the record you've just created. Read Dave Harms article [Introduction to SQL - Part 4](#) for background on the auto-incrementing problem.

There are different ways to solve the auto-incrementing problem; I will use the solution provided by Jim Kane (which is an adaptation of an initial solution by Scott Ferret). You can download Jim's code at ftp://www.icetips.com/pub/old_stuff/sqlan55.zip

From the downloaded file, follow these steps (culled from the readme.txt file) to install the code:

1. Copy `SQLAn.TPL` to the template directory and register `SQLAN.TPL`
2. Copy `SQLAN.CLW` and `SQLAN.INC` into the Clarion Libsrc directory
3. In `ABFile.inc` add two methods:

```
SetAutoIncDone Procedure(BYTE pAutoIncDone)

GetAutoIncDone Procedure(),BYTE
```

4. In `ABFile.CLW`, add the code for the two methods mentioned in 3 above. The source code is in `ABFix.CLW`.
5. Add the Global Extension to your application.
6. You are to create a table called 'dummy' in SQL Server. (The script to do this has been added to the example SQL Script.)

```
CREATE TABLE DBO."Dummy" (
  "dummy_col" INT)
```

Connection String

The Connection String is a string sent to a database server that allows your client machine to access the database. This is separate from the normal network rights given to you to connect to the Server. In order to understand how the connection string is used, it is important to understand the security features in SQL Server.

Security in SQL Server can be implemented using Windows Authentication Mode, or Mixed Security Mode.

- Windows Authentication works only on operating systems that use Windows NT authentication; you cannot use this with Windows 9x or Millennium servers. As clients, these operating systems must have a trusted connection to the Windows NT/2000 Server. This is because SQL Server allows connection based on the user account name or group membership available in the Windows domain by mapping logins from a trusted NT Domain into SQL Server.
- Mixed Security Mode: In mixed security mode, a user is given access by Windows Authentication or SQL Server authentication. If SQL Server authentication is used, then the password and username is maintained by SQL Server.

Remember when you are installing SQL Server to choose Mixed Mode (not Windows Authentication Mode). As you do this, you will be required to enter a password; this password should be `sa`.

The connection string is made up of the server name, the database you are connecting to, your username, and password. It is in the form:


```
'AYO2000-OFF, INV_SQL, sa, sa '
```

where AYO2000-OFF is the name of the Server, INV_SQL is the Database name, sa is the username, and sa is the password.

If SQL Server is running on your local machine, you can use this connection string:

```
'(LOCAL), INV_SQL, sa, sa '
```

SQL Server 2000 was released with support for multiple instances. This means you can run multiple copies/different versions of SQL Server 2000 on the same machine at the same time. If you install SQL Server using the instance feature you have to qualify the server name. So, assuming you created an instance called AYO_NV01 during setup, and your machine/server name is AYODAHUNS', then your connection string will look like this:

```
'AYODAHUNSI\AYO_NV01, INV_SQL, sa, sa '
```

From way the username and password is passed with the connection string, you can see that there are obvious security issues. For now, I will stick with this approach. In upcoming articles, I will explore more secure ways of logging on to the server.

Create a Global variable called GLO:ConnectionString to store the login information. Make it a CSTRING(128). Assign a value to this variable before the Open Files embed in the Main Frame:

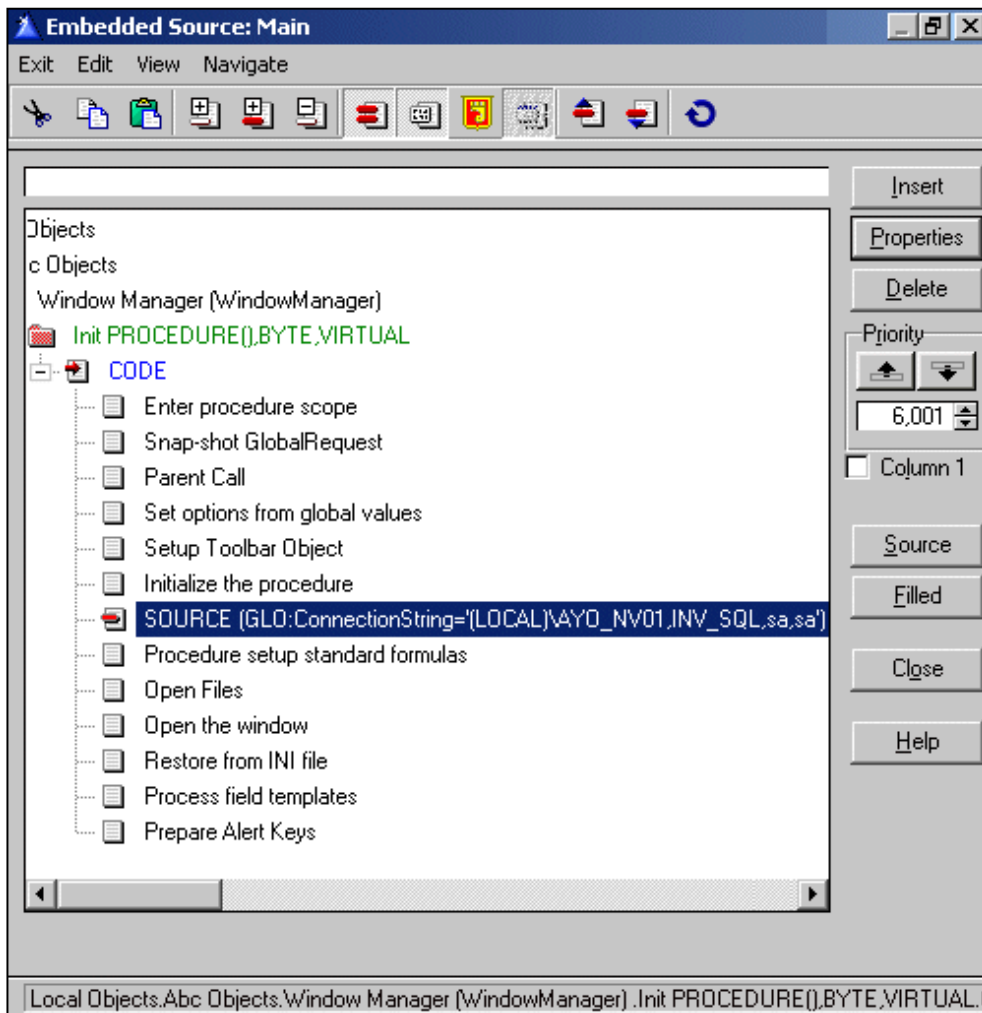


Figure 5. Embed point for connection string.

This connection string is hard-coded in this example, but in a real-life situation, you can read this from an INI file, or, use a more secure means of building the connection string.

You now know how to set table properties, create identity fields, install the auto-incrementing code, and create a connection string. In [Part 2](#) of this series I'll explain how to go about creating your tables on the server and getting your SQL data to work properly with Clarion.

[Download the source](#)

Resources	
Professional SQL Server 7	Robert Viera
SQL Server 2000 Developers Guide	Micheal Otey, Paul Conte

The Guru's Guide to Transact-SQL	Ken Henderson
SQL Server 7.0 Administrator's Pocket Consultant	
SQL Team	http://www.sqlteam.com
Microsoft SQL Server Home page	http://www.microsoft.com/sql/default.asp
Help, articles, and scripts for SQL Server	http://www.swynk.com/sql/
Interactive Product Guide	http://www.winntmag.com/Techware/InteractiveProduct/SQL2000/
SQL Server Magazine	http://www.sqlmag.com

[Ayo Ogundahunsi](#) presently lives in Henderson, Nevada, about ten minutes from Las Vegas. He works for [Impac Medical Systems Inc.](#), the leading company in cancer therapy software (written in Clarion). Impac has its headquarters in Mountain View, California. Ayo is married to Ayodola, and they have two boys, Darren and Joshua.

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 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

Using Example Files With TPSFix

by **John Heck**

Published 2001-08-10

I had been working with an application and the Employee file became corrupt. So I did the normal run of TPSFix and selected the corrupt Employee file to fix. After TPSFix completed I deleted the corrupt file and renamed the Employee.TPR file to Employee.TPS. The file worked fine for a while and then it became corrupted again. After a point in time it became so corrupted that I could not even view it through the TPS scanner.

I found that there is a way to resolve this kind of problem when using TPSFix. The first step is to create an empty TPS file via the application dictionary. So for this example you would create an empty Employee.TPS file. Now rename the file to Employee.TPE; this file will be used on the second screen of the TPSFix process.

Start TPSFix and complete the first screen with the name and location of the corrupt data file (see Figure 1). Accept the default 'Destination (result file)' and press the Next button.

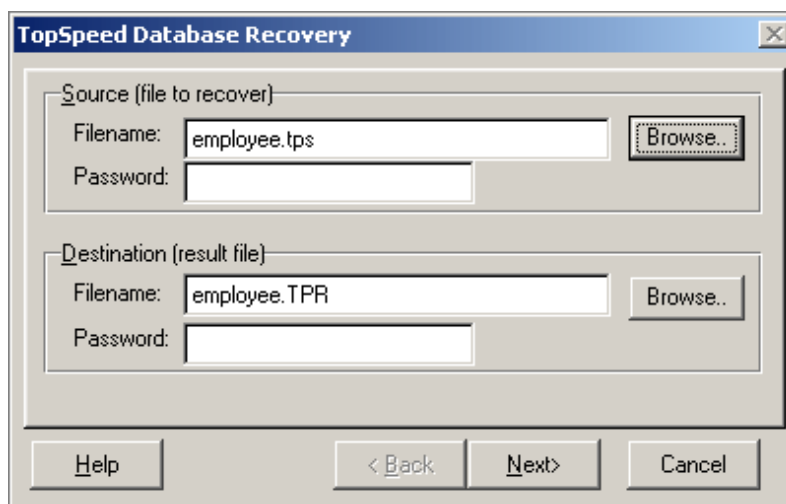


Figure 1. Specifying the source and destination files

The second screen allows the user to enter an Example File. This file is the empty Employee.TPS file that was renamed to Employee.TPE, as in Figure 2.

[Search](#)

[Home](#)

[COL Archives](#)

[Information](#)

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)



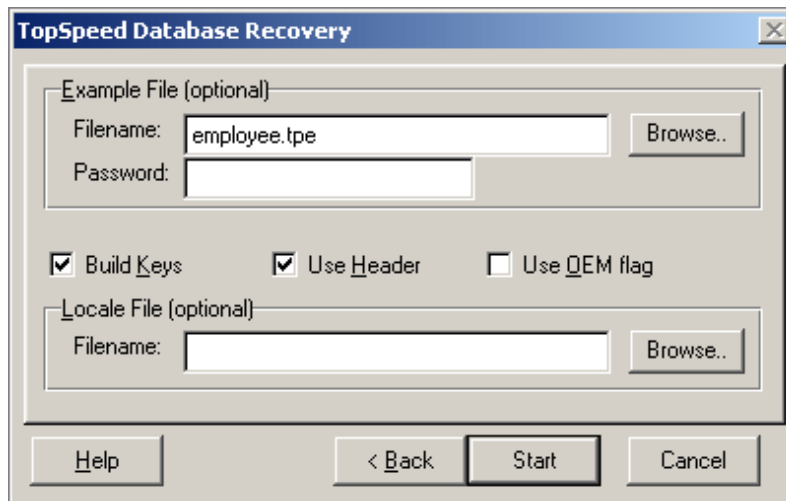


Figure 2. Setting the example file

When you use the Browse button to select the example file, the file dialog uses the default extension .TPE. This example file is nothing more than a file with a good header definition record. TPSFix then uses this file to rewrite the header and data from the corrupt file, rather than using the header from the corrupt file, which could be causing the problem.

Press the Start button to start the fix process. The corrupt file should not be a problem any more after this process is complete.

[John Heck](#) develops and plans systems development projects for small medium and large companies, using Clarion versions 2.1 and 3.0/3.1 and Clarion for Windows 2.003, 4 and 5/5.5 Enterprise Edition. He is also experimenting with Clarion's web development capabilities. In his spare time John enjoys spending time with his wife, gardening, playing with his two malamutes Justice and Matsi, traveling, and reading.

Reader Comments

[Add a comment](#)

**[John, If you renamed your corrupt file first - from...
You don't need to rename anything: Change tpsfix to...
Have the dictionary create the .tpe file by just entering...](#)**

Copyright © 1999-2001 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

Internal Help - An Alternative To Commercial Help Systems

by **Tim Phillips**

Published 2001-08-09

Providing Help in my applications is something I've struggled with for quite a while. I started with no Help (how to run my software is self-evident!) and then moved on to Windows standard help files for a few applications. This worked OK, but it wasn't the best solution for me for three main reasons.

The first reason I encountered was that writing the Help itself was work, but remembering how to use my Help Generator from time to time was a real pain. I was constantly forgetting the exact correct sequence of steps and pieces needed to add a new Help item. The more frustrating it was, the less I wanted to write Help. And the more time I spent fighting my Help Generator, the less actual Help got written.

Going along with this was the headache of remembering the Help ID that I had just created for a Help button in Clarion, which I had to duplicate perfectly when making up a Help file or the Help Topic wouldn't appear when the button was pressed. This was a real pain when I was working fast, as I could create dozens of Help IDs in an afternoon, and then I'd have to debug them one at a time to get them all working.

The final straw was when my applications began to be run from a CD or from a Terminal Server. Far too often my programs didn't seem able to correctly locate the needed Help file when it was called. How could I tell a user to read the Help when it wouldn't even open correctly?

A simpler solution

I needed something simpler. Ideally, I needed a way of displaying Help that was literally created with Clarion and thus as easy for me and my staff to use as my own applications.

As I stared at an collection of Help files one day, I realized that at its most basic, Help is just a display of a specific block of text

[Search](#)

[Home](#)

[COL Archives](#)

Information

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

Downloads

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)



when a button is pressed. This is a lot like displaying the memo of a record upon demand, something that Clarion can do easily.

From this understanding evolved what I call Internal Help. This is a simplified Help architecture which requires a single data file, a browse/form combination, a report, and two display windows. This architecture provides a window showing a memo full of text (the Help) which is displayed when the user presses a Help button. It also provides an easy way for all the Help in the system to be browsed and edited, and printed out upon demand.

The single data file is a Topspeed file (my preferred format) called InternalHelp. It has a record with a HelpTag field (a string of 40), a HelpTitle field (a string of 100) and a HelpMemo field (a memo of 64000) and a prefix of INT. There are two different unique keys, one on the HelpTag field (called ByHelpTag) and an ascending unique key on the HelpTitle (called ByHelpTitle).

In the application I want to add Internal Help to, I started by creating a standard browse/form combination to use in maintaining the InternalHelp File.

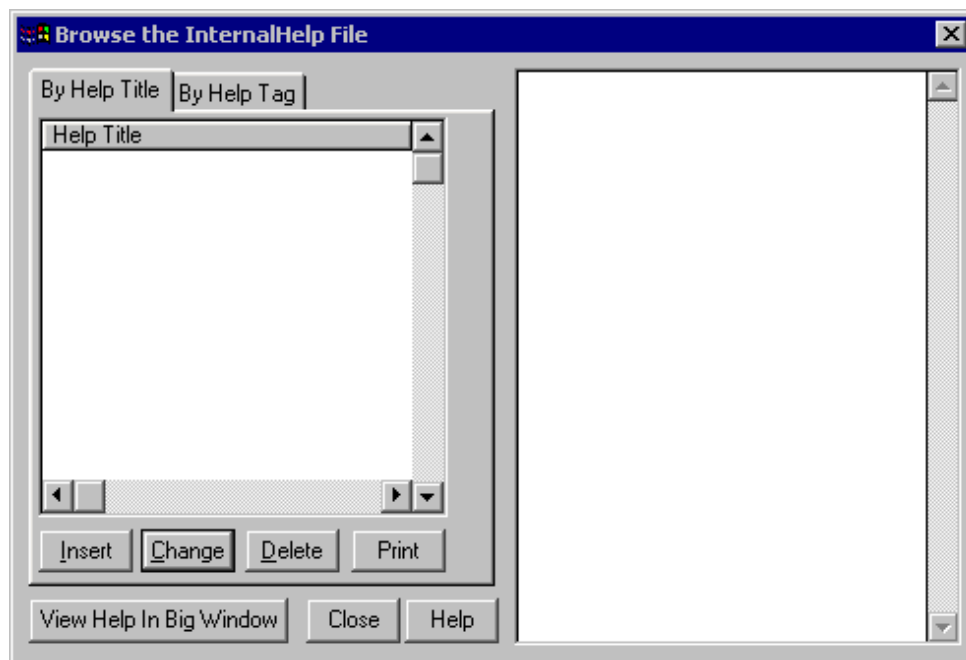


Figure 1. The InternalHelp browse

The BrowseInternalHelp procedure's primary sorting order is ByHelpTitle. ByHelpTag is a secondary sorting order, which is really only useful to a person actually writing help (to find where you've already used a Help Tag) so I generally HIDE() this tab from anyone who doesn't have my user ID. The browse itself has HelpMemo defined as a hot field (so the user can see the Help Description as the browse is scrolled), and the HelpMemo field's Read-Only and SKIP checkboxes are set on. The Print button calls the PrintHelpItem procedure, and the View Help In Big Window calls the ViewHelpInBigWindow procedure.

The `PrintHelpItem` procedure is a standard Clarion report which has `INT:HelpTag` as the Range Limit field with a Range Limit Type of Current Value (these settings will make it print only the highlighted Help item). Remember that the `HelpMemo` field needs to be defined as one line tall (make it match the height of the `HelpTitle` field) and make sure `Resize` is checked off on the Extra Options tab for the `HelpMemo` field so it will print the entire contents of the memo. This will allow the users to get hardcopy of any Help that they desire.

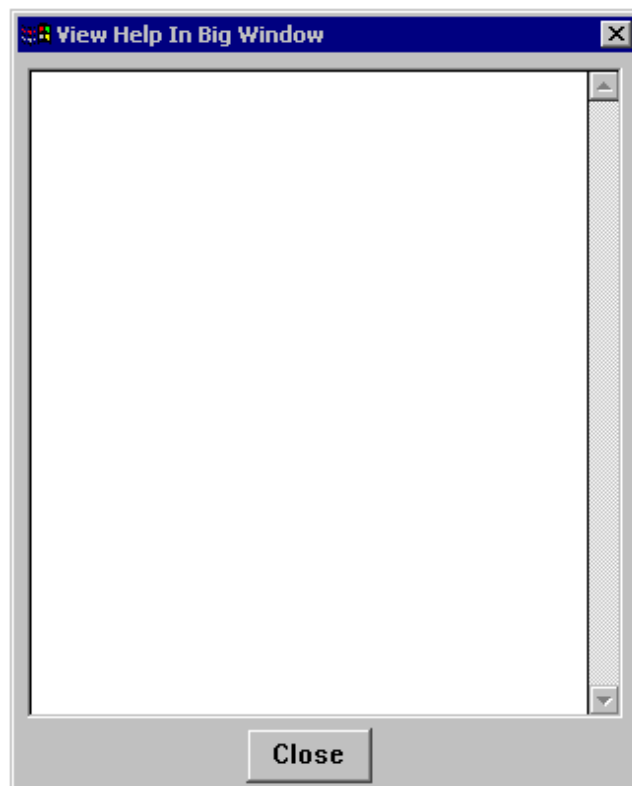


Figure 2. The ViewHelpInBigWindow window

The `ViewHelpInBigWindow` procedure is just an MDI child window set to cover the screen vertically as much as possible so the memo can be read without scrolling down (or with a minimum of scrolling). Put the `InternalHelp` file into Other Files in the procedure's file schematic, and put `HelpMemo` onto the window. I've found you don't want to make the `HelpMemo` text field any wider than on the `UpdateInternalHelp` form; you just want to make it taller so you can see everything (don't forget to include a vertical slider for those really long Help descriptions). I generally put a Close button on this window since having just the X in the upper right corner is too subtle for some users.

On the application's Frame, I use the Hide option for the default Help entry on the menu (so I always have the option of including Windows Style Help in the future with minimal trouble) and I add in my own Menu item that reads Help. This item's action is set to call the `BrowseInternalHelp` procedure with the `Initiate Thread` box checked. This means that when users select Help on the main menu, they get the browse of all the Help items. They can scroll

down through them and view/print any individual Help record that they wish. A user with the proper access permissions in my home-brewed security system can get access to the Browse's CHANGE button and alter the memo of any record (and thus easily alter what the Help record says).

I can make general purpose entries (Getting Started, Guide to Help, Vocabulary) appear at the top of my listing by just inserting a blank space or two at the front of the HelpTitle field (so it appears indented), and the normal sorting order of an ascending string key will put them right at the top where they are the first things that the user sees.

The application now has Clarion standard tools to maintain the data for the Internal Help system. The final programming step is to devise a way to display a particular Help record when a Help button is pressed. I chose to do this by co-opting the normal Help button that Clarion Wizards place on any browse or form window. I go to the properties window for the button and erase the STD ID field on the Extra tab.

I then go to Actions and set the action When Pressed to Call a Procedure, and provide a Procedure name of ViewHelpWindow. I leave the Initiate Thread field unchecked and in Parameters I put an entry like 'BrowsePeople' to identify the HelpTag I wish to have the user view.

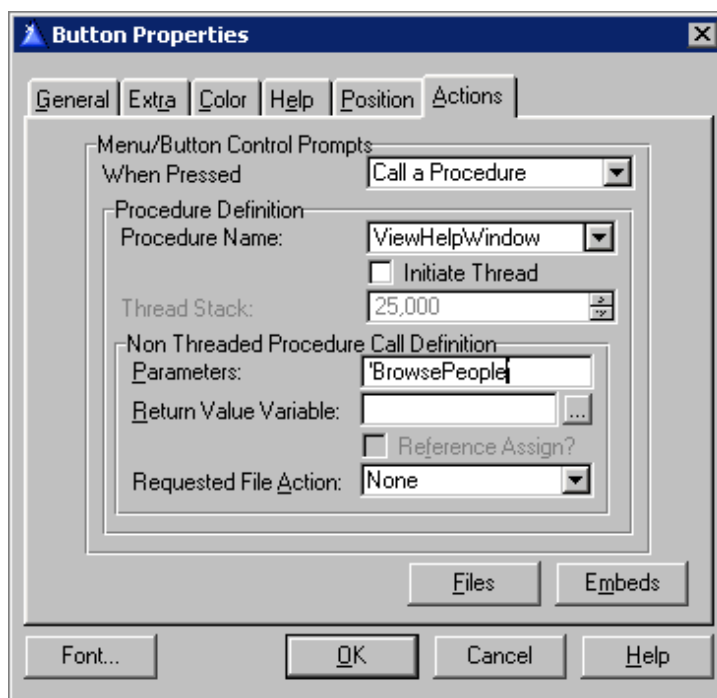


Figure 3. Setting the Actions tab for a help button

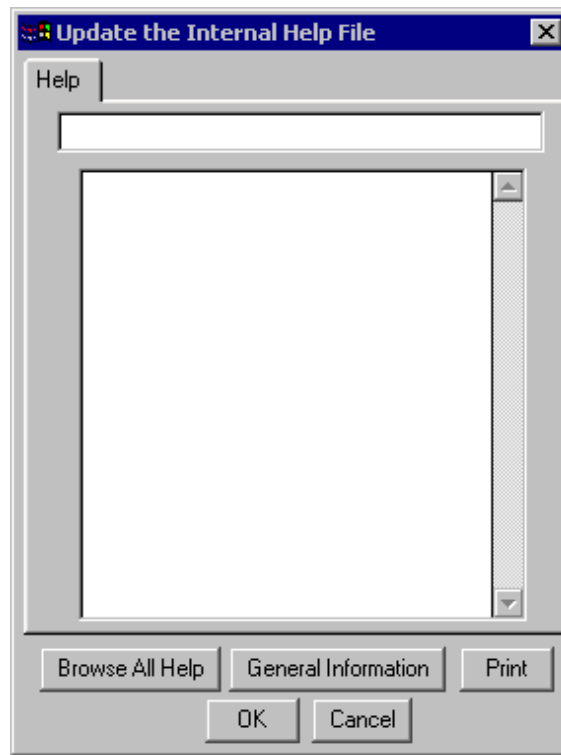


Figure 4. The ViewHelp window

Build the ViewHelpWindow copying the UpdateInternalHelp form to the name of ViewHelpWindow. By doing this, you get all the code architecture so you can update a Help item right from the Help window itself. So long as you are in ABC code, you can go to the ThisWindow.Initiate embed after the file is open and add the following code, which will attempt to fetch the correct Help record from the database, and then initialize fields for the Help record if the fetch fails so you can enter the Help right from the ViewHelp window:

```
INT:HelpTag=HelpTag
If Access:InternalHelp.Fetch(INT:ByHelpTag) |
  <>Level:Benign
  Self.Request= InsertRecord
  INT:HelpTag=HelpTag
  INT:HelpTitle=|
    ' No Help Entered For This Button'
  INT:HelpMemo=' Maybe the General Help ' |
    & 'Information or Browse Help Button' |
    & 'contains the information you need'
else
  Self.Request = ChangeRecord
END
```

The HelpTag variable is the parameter which is passed in by the calling procedure. This code will either fetch the matching Help record and put the user into CHANGE mode or (if the Help record doesn't exist) it will put the user into INSERT mode and default the field contents so the new Help record can be made up. (WARNING: if you put this code in, remember to include some security, or anyone viewing Help could change it on you.)

At the bottom of the `ViewHelpWindow` window I put three new buttons. The **Browse All Help** button calls the `BrowseInternalHelp` procedure on its own thread (this gives the user access to all the Help if they have hit a Help button anywhere in the application). The **General Information** button calls the `ViewHelpWindow` procedure and passes in the parameter of 'GeneralHelp' (which is a Help record that describes how to use the Help system and some general information about how to use browses and forms in Clarion). The **Print** button calls `PrintHelpItem`.

That is the whole system. I can now go through my application redirecting any Help button to `ViewHelpWindow` with an appropriate `HelpTag` as the input parameter. After recompiling the application, I can run it and go to each Help button and press it, fill in the Help Title and Help Memo and press OK to save the record. Quickly and easily, I can write all the Help I need. I can rewrite any Help record from the `ViewHelpWindow` or from `BrowseInternalHelp`. To update the Help my staff see, I just replace the copy of `InternalHelp.TPS` and they are current.

So, how close have I come to matching my original goals?

Did I make Help easier to write? Now I don't need anything but my own application to maintain Help text. Creating a new Help item is as easy as mildly modifying a Help button. The whole Help system is in Clarion and built with components that I use all day and have a lot of comfort/experience with using.

Did I get rid of having to remember Help IDs? It is still possible to make a mistake and reuse a HelpID, but by using HelpTags based upon the name of the window the Help is for, I don't have that happen often. I don't have to remember the HelpTags in detail at all. After adding new Help calls in a program and compiling, I can go through the program to each Help button and it will automatically populate the requested HelpTag when Help is pressed and none is found. I just fill in the Memo and Title and save the record, and the Help is loaded.

Did I make the Help easier to run from Terminal Server and various installation locations? Internal Help is just part of the application. If I can get the drive-mappings and other things to work for displaying any data with a Clarion application, I can display Help with the exact same techniques. This has made deployment much easier in my experience.

Summary

I think this Internal Help scheme works pretty well, but what I have described is a little more spartan than the help system I normally use. I typically add my own home-brewed security templates to control ability to change the data (any of the third party security templates should do equally well). I also add Brian Staff's [XPLORE](#) template to the `BrowseInternalHelp` procedures

to make finding records easier, and I'm looking at [SearchFlash](#) to provide a way for users to search through the text of the Internal Help records.

I've thought about building a "prettier" user interface using the formatting options available with the Clarion 5.5 RTF control, or maybe generating a cluster of interlinked HTML files that would be the Help for an application on an internal website (for some reason, people will actually spend hours looking for a website and reading its contents while ignoring a Help file only a mouse-click away with the same information in it). Since the entire Help system is just TopSpeed data delivered by Clarion code, the limits are the creativity of the programmer in displaying and managing the Help file data itself.

[Download the source](#)

Tim Phillips began programming Clarion with Version 3.0 for DOS. He currently works mainly with Clarion 5 but is determined to get everything into 5.5 by 2002. His preferred programming technique involves a lot of bass rock guitar on his cordless headset and vigorous application of the Keep It Simple Stupid principle. When not programming, he can be found trying to write fiction, "building Legos" with his two nieces, or watching too much quality television.

Reader Comments

[Add a comment](#)

You might gain some productivity by creating an Internal...
I'm sorry but this is nonsense. You have given your user...
James, I thought you said the best help was no...
James, You do raise some valid points, some of which I...
If your users can't use help, don't give them...

Copyright © 1999-2001 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.

INVEST
in your own abilities

Clarion
magazine

published by
CoveComm Inc.

Clarion MAGAZINE

Understanding Recursion - Part 2

by **Dennis Evans**

Published 2001-08-08

[Last week](#) I explained the basics of local scope, stacks and procedure calls. Now it is almost time to investigate recursion. First, however, I'd like to clear up a common example from Clarion that is *not* recursion.

Iteration, not recursion

When building an update form, using the standard templates, there are two options available concerning the action when the user presses the Ok button. The first option, Return to Caller, simply adds one record to a data file and returns to the calling procedure. The second option clears the record buffer and allows the user to add additional records. The number of records added, without leaving the update form, is decided by the user at runtime. The user may add 2, 3, 4 and so on. This is often referred to as recursively adding records. This is actually an example of iteration, not recursion. The call to the update procedure is not recursive, and the action is being done to the same object. The update procedure is simply looping X number of times.

Recursion

One of the common examples used in teaching programmers recursion is calculating the factorial value of an integer. This example is popular because the differences between a recursive algorithm and an algorithm that uses iteration can be shown in a few lines of code. Both are relatively simple and the solutions may be easily compared.

According to Webster's dictionary, the factorial of a number is "the product of all positive integers from 1 to N." Factorials are written as N!. If N is a positive integer then N! can be written as

$$N * ((N - 1) * (N - 2) * (N - 3)) \dots$$

A slightly shorter version could be written as:

[Search](#)

[Home](#)

[COL Archives](#)

Information

[Log In](#)

[Membership/](#)

[Subscriptions](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

Downloads

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Reader](#)

[Comments](#)



$$N! = N * (N - 1)!$$

Take a moment and examine the shorter version. Note that one is subtracted from the value of `N`. If you were to write a procedure to solve this problem, each call would use a smaller value of `N` than the previous call. Part of the definition of tail recursion is each recursive call uses a smaller piece of the object, in this case a smaller number `N - 1`. In other tail recursive procedures you might be passing the recursive call a smaller part of an array or some other data structure.

Now take a look at the code to calculate the factorial of a positive integer. The first version uses iteration and the second is the recursive version.

```
FactorialIteration procedure(long N),long
Factor long(1)
Count long,auto
  code
  loop Count = 2 to N
    Factor = Factor * Count
  end
  return Factor
```

```
FactorialRecursion procedure(long N),long

  code
  if (N = 0)
    return 1
  else
    return N * FactorialRecursion(N - 1)
  end
```

The version that uses iteration has two local variables, `Factor` and `Count`. The recursive version does not use local variables. Recursive algorithms may use local variables, but a recursive version of an iterative procedure will always use fewer local variables. As well, an iterative version will always have a loop structure as its main control, while a recursive version will always have a conditional branch statement as its main control.

The iterative version creates a local variable `Factor` and initializes it to the value of 1. The procedure then loops from 2 to `N` times, multiplying the value of `Factor` by the loop control variable, `Count`, and storing the result back into `Factor`.

The recursive version is a little more interesting. If you are not familiar with recursion it may not be apparent how the procedure works or that it will even produce a result, but the recursive version will produce the same result as the iterative one. I'll explain how it works shortly.

One last comment on the two procedures. In a working application you would not use the recursive version, because the iterative version is faster and simpler to code. As I said earlier, this is a

teaching example. I will take up the subject of when to use recursion later.

Recursive calls

In the iterative version of the factorial problem the procedure is called once. Internally the procedure loops from 2 to N times. In the recursive version the procedure is called $N + 1$ times with each call performing the process outlined above. When the procedure that uses recursion is called it will evaluate the parameter N . If N is equal to zero the recursion stops and a value of 1 is returned ($0!$ is defined as 1).

If N is not equal to zero, the `IF` statement will fail and the `ELSE` clause will execute. When this clause executes, the value of N is multiplied by the value returned from the recursive call. Remember the recursive call happens first, then value of N is multiplied by the returned value. Because of the scope of local variables and parameters these subsequent calls to the procedure are not aware of the current value of N . Each time, the procedure is called N is reduced by one, so each call of the procedure is dealing with smaller values of N . Early in the process of a procedure call the caller evaluates the parameters, in this case ($N - 1$). The caller does not push the current value of N and a 1 on the stack. The program will perform the subtraction and result will be pushed onto the stack.

A diagram and step by step trace of the calls makes recursion easier to follow. Figure 4 shows the recursive calls. In the example, the first function call looks like this:

```
FactortialResult = FactorialRecursive(4)
```

Program execution then moves to the `FactorialRecursive` procedure, after a stack frame has been created.

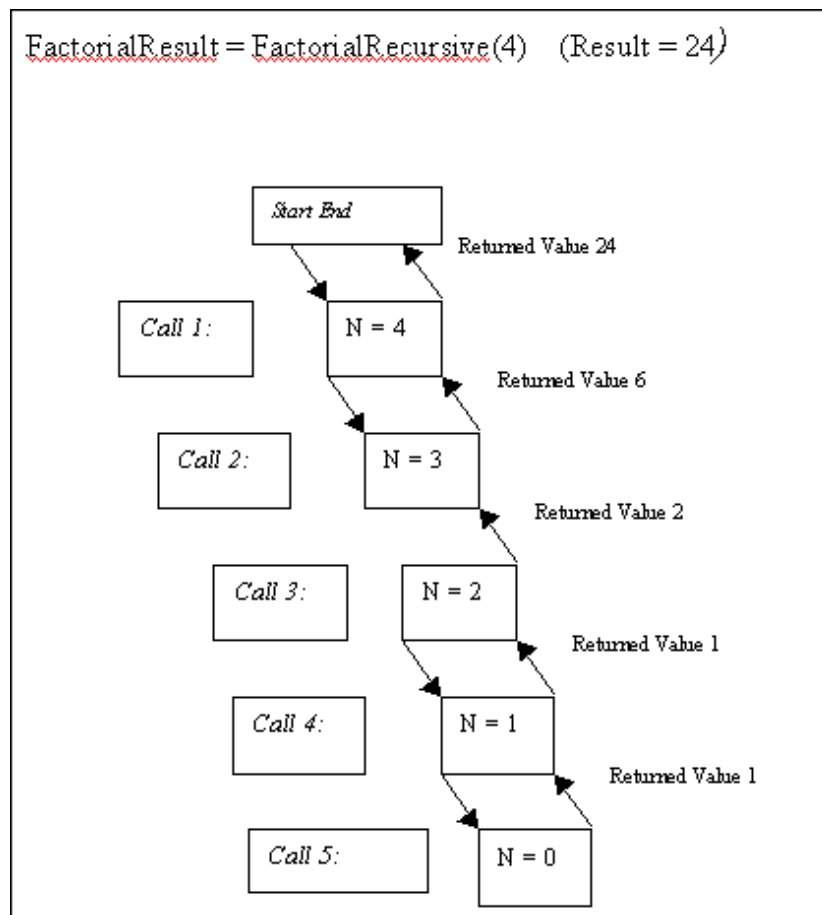


Figure 4. The recursive procedure calls

Following Figure 4, first trace the calls down. Each time the procedure is called a new and separate instance of N is created locally. This local instance of N is visible only inside the procedure it was passed to as a parameter. On each of these calls the value passed will be $N - 1$.

Call 1	Value of N is 4, not equal to 0, the IF clause fails and the ELSE clause is executed, return $N * \text{FactorialRecursion}(N - 1)$,
Call 2	Value of N is 3, not equal to zero, the IF clause fails and the ELSE clause is executed, return $N * \text{FactorialRecursion}(N - 1)$
Call 3	Value of N is 2, not equal to zero, the IF clause fails and the ELSE clause is executed, return $N * \text{FactorialRecursion}(N - 1)$.
Call 4	Value of N 1, not equal to zero, the clause fails and the ELSE clause is executed, return $N * \text{FactorialRecursion}(N - 1)$
Call 5	N is zero. Now the IF clause succeeds and the value of 1 is returned.

Follow the execution chain of the five calls down, several times if needed, until you have a clear idea of what is happening, what the values of N are and why. Next the sequence will reverse; if you have any confusion about the downward movement, understanding what is happening moving back up chain will be difficult.

Call 5	N is equal to zero so the IF statement succeeds and a value of one is returned to the caller, call number 4.
Call 4	Call 5 has completed and returned a value of 1 to this procedure. One is multiplied by the value of N, which is 1. 1 will be returned to call number 3.
Call 3	Call 4 has completed and returned a value of 1 to this procedure. One is multiplied by the value of N, which is 2. 2 will be returned to call number 2.
Call 2	Call 3 has completed and returned a value of 2 to this procedure. Two is multiplied by the value of N, which is 3. 6 will be returned to call number one.
Call 1	Call two has completed and returned a value of 6 to this procedure. Six is multiplied by the value of N, which is 4. The value of 24 is returned to the original calling procedure and the value is assigned to the variable <code>FactortialResult</code> .

Again follow the chain up until what values are returned and why is clear. If you are comfortable with the above sequences then you have recursion call chain beat! If you are not comfortable or unsure of what the code is doing, take some time and work through the chain until you are completely comfortable. If you need to work through the process several times then do it. There is nothing magical or even complex about what is happening, but it is a somewhat abstract concept. Take the time to follow the chain along with the text, and remember each time the procedure is called a new instance of the procedure is created. Understanding will eventually happen.

I can assure you from experience that understanding will not come gradually. What will happen is that after following the chain up and down several times the process will not make any sense. Then you will wonder what kind of black magic is in use and start thinking 'I really don't need this aggravation.' But because all programmers are stubborn you'll attempt one last trip down the chain. That will be when it hits. Somewhere, somebody turned on the switch and you will understand a recursive call chain.

There is one more important recursion topic: the base case. The base case is a condition used by the recursive procedure to indicate when to stop making recursive calls. In the factorial example the base case is `IF (N = 0)`. When you understand the recursive example above and the base case, the rest is deciding

when to use recursion and when to use iteration.

In an upcoming article I will examine the base case and show a couple of methods to help prove the base case. I'll also provide some additional examples of recursion and explain when to use recursion and when to use iteration.

[Dennis E. Evans](#) is retired from the U.S. Army. During his time in the military he spent twelve years in the Armored field and eight years in information management. He currently works as an independent contractor and resides in Marion, Illinois with his wife Beverly and their two children Christopher and Jessica. His hobbies include historical simulations, reading and studying different programming languages.

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 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.