

Clarion Magazine

Reborn Free



Elvis To Perform At ETC 2004

Word has leaked out that Elvis Presley is alive and well, and will be performing at the Edgewater Hotel in Gatlinburg, TN, during the Tuesday night reception for Lee White's etc 2004 Clarion conference!

Posted Thursday, April 01, 2004

PDF For March, 2004

All Clarion Magazine articles for March, 2004 in PDF format.

Posted Monday, April 05, 2004

Tips & Techniques Book 15% Off Sale Ends Friday!

The 15% off introductory special on the Tips & Techniques book ends April 16. Order now before the price goes up!


Posted Wednesday, April 14, 2004

Take Our Book Survey

The Clarion Tips & Techniques book (15% discount ends Friday!) has been a terrific success, and we'd like to produce more books like it. But with seven years worth of articles to choose from, there are a lot of possibilities. We need your help to know which material would be of most interest to Clarion developers. This survey will only take a minute or two of your time.

Posted Wednesday, April 14, 2004

Articles: 

News: 



News

[Clarion 6.1 Ready](#)

[Lindersoft Under Spam Attack \(April 30\)](#)

[Jaguar Generates Java Apps](#)

[xDataBackup Manager Pro 1.8](#)

[Adi0Cornerz Update](#)

[Wi-Fi WEP Key Generator](#)

[AdiIniEditor Upgraded](#)

[CWPlus 3.00 Special Offer Expires Soon](#)

[etc2004 Open Sessions](#)

[CPCS ClarioNET Previewer For C6](#)

[New 1st Icon Design Bundle](#)

[New Firebird Forum](#)

[PlugIT 2.0 Available At RADFusion](#)

[SB5 Beta Registration Codes](#)

[WPViewPDF Control Wrapper](#)

[Clock With Mouse 1.01](#)

[EasyExcel 3.04](#)

SURVEY

Where on the screen do you dock the Windows taskbar?

Bottom (Windows default)

 92.8%

Top
 4.8%

Left side
| 1.2%

Right side
| 1.2%

83 responses

[Previous Surveys](#)

One Year Ago In CM

[Clarion 6 EA4 Released](#)

[Data Structures and Algorithms Part XX - Topological Sort](#)

[Data Structures and Algorithms](#)

A Hunka-Hunka Burnin' CD Data, or Two Steps Forward, One Step Back(up)

An integrated backup system can be a great way to provide complete service to your customers. But what happens when your users want to back up to CD? Mark Riffey explains a technique that makes use of the Windows XP burn wizard.

Posted Thursday, April 15, 2004

The Shortest Useful Clarion Program

How little code can you write and still have a broadly useful program? As Tim Phillips found out, sometimes just a couple of lines of Clarion code can do wonders.

Posted Friday, April 16, 2004

Book Review: SQL Tuning

Among Clarion developers SQL databases continue to grow in popularity. It's easy to see why - SQL offers vastly increased capabilities over flat file databases. But with power comes complexity, and with complexity comes the possibility that your SQL queries may not be running as quickly as they could. Tuning SQL statements often seems more black art than science, but in this book Dan Tow introduces a simple and effective methodology, and applies it to a variety of common problems.

Posted Friday, April 16, 2004

Weekly PDF for April 11-17,

[etc2004 Only Seven Weeks Away](#)

[xWordCOM Docs Beta 1.2](#)

[Online "Why Clarion" Demo](#)

[JaduTech's New Home](#)

[AdiTech Templates](#)

[Developer's Conference In Brazil](#)

[CWPlus 3.00 Released](#)

[Icon Collection XP](#)

[Clarion Training Web Site](#)

[Public Clarion Bug/Workaround Repository](#)

[ABC Free Templates and Tools Updated](#)

[xInactivity v1.2](#)

[ThinkData Web Store Back Online](#)

[Britney Spears at etc2004!](#)

[Castle Computer Moving Sale](#)

[Yet Another Clarion Wallpaper](#)

[SQL Script Site](#)

[#cw-talk Clarion IRC Chat](#)

[The Fourth, And Last, ETC](#)

[etc2004 SQL Topic Poll](#)

[ThinkData Server Migration](#)

[BST 3.0 Released](#)

[UK Clarion News](#)

[CPCS Support Schedule](#)

[Bug Poster 2.0 Available](#)

[cpTracker Pro 2.0 Now Available](#)

[Part XIX - Simple Graphs](#)

Two Years Ago In CM

[SoftVelocity's Upcoming Product Technology - Abstract](#)

[SoftVelocity's Upcoming Product Technology \(Part 1\)](#)

[SoftVelocity's Upcoming Product Technology \(Part 2\)](#)

Three Years Ago In CM

[Clarion News - May 2001](#)

[Replicating IDLE: All Quiet on the Keyboard?](#)

[The Clarion Advisor: API Tricks](#)

Four Years Ago In CM

[Clarion: Back Where It Belongs](#)

2004

All articles for April 11-17, 2004 in PDF format.

Posted Wednesday, April 21, 2004

Postmortems and The Clarion Tips & Techniques Book

Andrew Guidroz II tries to learn a little something from every day's work. A trip to a client site, some unexpected code copying, and a few articles from the Tips book add up to a useful postmortem.

Posted Thursday, April 22, 2004

Basic File Handling in Clarion

Over the years, Dermot Herron has developed a set of basic practices for hand-coded file handling using Clarion's ABC library. In this article, based on a talk he gave to the Johannesburg Clarion User Group, Dermot explains his tried and true approach.

Posted Friday, April 23, 2004

Understanding Clarion Templates, Part 3: Introducing Code Templates

The Clarion template system is, at heart, a simple idea. If you want to generate code, based on varying conditions and requirements, you just create a template programming language that lets you generate Clarion language statements. Okay, maybe the reality isn't all that simple, but the easiest place to start understanding the template language is with code templates, as David Harms explains.

Posted Friday, April 23, 2004

[solid software Holiday Schedule](#)

[DevCon 2004 Call for Papers](#)

[Buggy Version 4](#)

[DevCon 2004 Highlights](#)

[xWhatsNew Class 1.4](#)

[ClarionPost Product Listings](#)

[ClarioNET For Clarion 6](#)

[xRuntimeStyle Manager 1.6](#)

[Search the news archive](#)

[Template Viewing Utility](#)

[The Cranky Programmer: Survey This!](#)

Weekly PDF for April 18-24, 2004

All articles for April 18-24, 2004 in PDF format.

Posted Tuesday, April 27, 2004

Burning COM: How To Write CDs in Windows XP With ICDBurn

When Andrew Guidroz's article idea was made redundant by Mark Riffey's article on popping up the Windows XP CD burn wizard, Andrew decided to grab COM by the horns and figure out just how to use the ICDBurn COM interface in Clarion. He ended up with two examples, one using Jim Kane's COM classes, the other using SoftVelocity's COM classes.

Posted Thursday, April 29, 2004

Generating MS SQL Server Side Triggers

Although Ayo Ogundahunsi finds the new client side triggers bold and ingenious, he still prefers the traditional way of using server-side triggers. In this article he demonstrates how to use a custom template to create server side triggers for database auditing.

Posted Friday, April 30, 2004

Looking for more? Check out the [site index](#), or [search the back issues](#).

This site now contains more than 700 articles and a total of over a million words of Clarion-related information.

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Clarion Magazine](#)



Clarion News

[Search the news archive](#)

[Clarion 6.1 Ready](#)

Clarion 6.1 is complete, and will be available for download the week of May 4. Changes include XP theme support (PROP:ThemeActive); Win98/WinME bug fixes/workarounds; RTF embedded objects; new ErrorStatusClass; Oracle cursors in stored procedures; runtime DLL mismatch checking; new TransactionManager class; PDF encryption and compression; new HOWMANY, GETGROUP and ISGROUP language statements; XML BLOB support (base64); and several hundred other fixes, changes, improvements, and workarounds.

Posted Friday, April 30, 2004

[Lindersoft Under Spam Attack \(April 30\)](#)

Lindersoft is reporting a spam attack on its server. Responses to support requests may take longer than usual due to the flood of bogus emails.

Posted Friday, April 30, 2004

[Jaguar Generates Java Apps](#)

Softmasters has announced the beta release of JAGUAR, the first Clarion template chain that allows you to generate multi-platform applications. You can generate your applications once, and then run them on Windows, Linux, Mac, and many other platforms. The JAGUAR package includes: procedure templates, including Browse, Form, Process, Window, etc.; Application, Browse & Form Wizards that will allow you to create a Java application in minutes from your

Clarion DCTs; The JBCL, a library of classes on which all JAGUAR applications are based; The JCSL: a library of Clarion statements implemented with the only goal of shortening the learning curve needed to write Java applications. Product documentation is still in progress, but for those of you who can't wait this is an Early Release of JAGUAR v1.0 beta 1. You can order it now and save almost half the price of the gold release! The documentation and any product upgrades will be sent to the Early Release users with no charge.

Posted Tuesday, April 27, 2004

[xDataBackup Manager Pro 1.8](#)

xDataBackup Manager Pro v1.8 has been released. Changes include: Two new options in global template; ManualBackup method changed to allow silent backup or backup by timer. Updated documentation, demonstration program and installation kits for Clarion 5, Clarion 5.5 and Clarion 6 are available.

Posted Tuesday, April 27, 2004

[Adi0Cornerz Update](#)

Adi0Cornerz has been updated with improved group and option handling to facilitate a centered prompt outside the box.

Posted Tuesday, April 27, 2004

[Wi-Fi WEP Key Generator](#)

Ben Brady has a new utility for WEB key generation and wireless router management.

Posted Tuesday, April 27, 2004

[AdiIniEditor Upgraded](#)

AdiIniEditor has been upgraded with the ability to delete all INI sections that only contain window position information. When windows disappear this is a nice feature.

Posted Tuesday, April 27, 2004

[CWPlus 3.00 Special Offer Expires Soon](#)

If you order CWPlus 3 by April 30 you will save 25% off the regular price. CWPlus 3 is now \$74 (the regular price is \$99). When you are

ordering you must enter this promotion code in order to get the discount: Ing3. NB. This offer is suitable for purchasing: a full license (not an additional license); by normal customers (not by distributors).

Posted Tuesday, April 27, 2004

[etc2004 Open Sessions](#)

There will be a number of free training sessions before, during, and after etc, including sessions on NetTalk, Replicate, Insight Graphing, Fenix ASP.NET Generator, and Whitemarsh's Metadata Management.

Posted Tuesday, April 27, 2004

[CPCS ClarioNET Previewer For C6](#)

CPCS has now released vCN1.2 of the CPCS ClarioNET Previewer add-on. Existing users can install this release using their current install codes.

Posted Tuesday, April 27, 2004

[New 1st Icon Design Bundle](#)

A new icon bundle is available now and you can get it at a reduced price for a limited time. Included: XP Collection; XP People Collection; Red Theme Collection; Yellow Theme Collection; Blue Theme Collection; Green Theme Collection; Silver Theme Collection. Total icons: 841. Total files: 36,582. Sizes included: 64x64, 48x48, 32x32, 24x24 & 16x16; Formats included: ICO, PNG, BMP, GIF. Bundle valued at \$800, now \$299. Free shipping.

Posted Tuesday, April 27, 2004

[New Firebird Forum](#)

There is a new forum for Firebird users available at the Tek-Tips web site.

Posted Tuesday, April 27, 2004

[PlugIT 2.0 Available At RADFusion](#)

PlugIt 2.0 is now shipping, and is available exclusively from www.radfusion.com. Just follow the categories for Plugware. PlugIT is an advanced leak checker for Clarion based programs that is simple to use. Comes with a complete manual describing how to use the

template based leak checker in your applications (including any third party products). PlugIT 2.0 benefits hand coders too. The manual and example project shows how simple this is to ensure you have no memory leaks. PlugIT 2.0 is completely thread safe and works with Clarion 5, 5.5 and Clarion 6.

Posted Tuesday, April 27, 2004

[SB5 Beta Registration Codes](#)

Lindersoft has begun to send out SB5 beta registration codes. If you are a SB5 beta tester and you have not received your code by the end of next week, please send an email to sb5@lindersoft.com.

Posted Tuesday, April 27, 2004

[WPViewPDF Control Wrapper](#)

Klarisoft has released WPViewPDF, a control wrapper and templates for the wpViewPDF component from wpcubed GmbH (<http://www.wptools.com>). WPViewPDF can be used to quickly display and print PDF files which were created with the wPDFControl (http://www.klarisoft.com/KSNews_127.htm) or with other libraries such as PDF-Tools. It also supports ANSI text extraction and text find + highlight functionality. Control is directly embedded in a Clarion window and does not require any Adobe(tm) products to be installed. Just one extra DLL needs to be distributed with your app.

Posted Tuesday, April 27, 2004

[Clock With Mouse 1.01](#)

With ClockWithMouse, you can enter a time with an analog clock popup.

Posted Tuesday, April 27, 2004

[EasyExcel 3.04](#)

EasyExcel 3.04 is now available. Bug fixes include: ActivateWorkbook method GPF; SetBorder method incorrect drawing with Linestyle:None style; Link error (Syntax error Unknown Identifier XLALIGN: CENTER); GetVersion method fixed Unresolved External error. Changes include: new parameters for ColumnWidth and OpenWorkbook parameters, SetChartFont added ability to change font of a different chart areas.

Changed templates include Init, OpenDoc, and PageSetup code templates. New templates include ShowChartOnWindow control template and PrintChartInReport control template. EasyExcel uses COM technology so you receive full control over Microsoft Excel from your Clarion application.

Posted Tuesday, April 27, 2004

[etc2004 Only Seven Weeks Away](#)

etc2004 will be the last "etc" conference so, if you've ever wanted to attend an "etc" gathering or know of someone who would like to attend, now would be the time. etc 2004 offers three days of in-depth, hardcore, discussions of timely subjects from some of the best our community has to offer. Following the conference, Bruce Johnson will be conducting a day-long class on Clarion Class Design. In addition, there are two special sessions to be held on Tuesday, prior to the conference. Bruce Johnson, from CapeSoft, will be holding a training session on NetTalk, Replicate & Insight Graphing. Also, representatives from RADventure will be providing a demonstration of their Fenix ASP.NET Generator. There are also a great many door prizes donated by conference sponsors, including three Fujitsu P5020G Notebooks.

Posted Tuesday, April 20, 2004

[xWordCOM Docs Beta 1.2](#)

A new beta release of the xWordCOM rev 1.2 documentation is now available for download.

Posted Tuesday, April 20, 2004

[Online "Why Clarion" Demo](#)

Ben Brady has put up an online demo showing the power of Clarion.

Posted Tuesday, April 20, 2004

[JaduTech's New Home](#)

Steve Bottomley's JaduTech web site, home of the greenbar template, among other things, has moved. Update your bookmarks.

Posted Tuesday, April 20, 2004

AdiTech Templates

AdiTech has released several templates. AdiIniFileEditor is for those still using INIfiles, and Adi0Cornerz has been commercialized to secure future maintenance. Both templates are available from ClarionShop.

Posted Tuesday, April 20, 2004

Developer's Conference In Brazil

V CONDEV will be held June 10, 2004 to June 12, 2004 in Porto Alegre city, at Rio Grande do Sul. Presentations include: OOP Clarion - Matias Flores, Soft Masters, Argentina; OLE and DDE - Juan Domingo Herrera, Soft Masters, Argentina; Crystal Reports and Clarion - Julio César Pedroso; Templates - José Roberto Pereira; The New Clarion Product - Juan Domingo Herrera & Matias Flores; The New Template ADO (Clarion 6) - Julio César Pedroso; PostgreSQL Functions and Clarion 6 - Alessandro Kretzer.

Posted Tuesday, April 20, 2004

CWPlus 3.00 Released

Ingasoftplus has released CWPlus 3.00. This version supports Clarion 5.0, 5.5, and 6.0. This is a fully 32 bit rewrite. New features include: Support for scrolling using mouse wheel in all windows containing vertical scroll bar, including editor (not supported on Windows 98); You can control CWPlus by using the System tray icon; Having selected a name of the procedure in PopBox and entered a symbol you can insert at once the identifier of procedure in the editor and get the information about the prototype.

Posted Tuesday, April 20, 2004

Icon Collection XP

1st Icon Design has released Icon Collection XP, which contains 140 icons in each of the following formats and sizes, with or without shadow, for a total of 5880 individual files: Sizes: 64x64, 48x48, 32x32, 24x24, 16 x16; Types: WIN Icons, PNG, BMP, GIF.

Introductory price \$99 (reg \$129).

Posted Tuesday, April 20, 2004

[Clarion Training Web Site](#)

Mark Sarson is setting up a web site for all things based around Clarion Training. This is intended to be a resource for both students and teachers throughout the Clarion community.

Posted Tuesday, April 20, 2004

[Public Clarion Bug/Workaround Repository](#)

Mark Riffey has set up a central, public, open repository for bug reports, resolutions, workarounds, etc. There are a lot of categories, primarily to make it easier for the community to find/post things specific to their needs (and for SV to find items of interest to a particular developer/QA person). The categories are not intended to match the newsgroups. Mark will likely be seeking "moderators" for each category area.

Posted Wednesday, April 14, 2004

[ABC Free Templates and Tools Updated](#)

Changes to the ABC Free Templates and Tools as of 04/13/2004 include: Section of BrowseCallsvsSimpleStringFilterFunction was missing after re-organization; Added fix provided by Alvin at Surework to TPTerminateEXE function.

Posted Wednesday, April 14, 2004

[xInactivity v1.2](#)

SealSoft's xInactivity v1.2 is now available. New in this version: Small changes in class - AddEvents method now has a return value; Changes in template - three code template for AddEvent, ChangeEvent and RemoveEvent have been added.

Posted Wednesday, April 14, 2004

[ThinkData Web Store Back Online](#)

The ThinkData web store is finally back online after being brought down by a surprise server migration.

Posted Wednesday, April 14, 2004

[Britney Spears at etc2004!](#)

No, Britney Spears will not be at etc2004 - at least not that Lee is

aware of. This is just a shameless attention-getting ploy. etc2004, which is just eight weeks away, offers Clarion developers three days of in-depth, hardcore, discussions of timely subjects from some of the best our community has to offer. Carl Barnes - True Threading in C6; Dave Harms - XML for Clarion Developers; Andrew Guidroz - Cajun Cookout; Russ Eggen - Template Writing; Shawn Mason - SQL; Andy Ireland - COM & dotNET in Clarion. Following the conference, Bruce Johnson will be conducting a day-long class on Clarion Class Design. In addition, there are two special sessions to be held on Tuesday, prior to the conference. Bruce Johnson, from CapeSoft, will be holding a training session on NetTalk, Replicate & Insight Graphing. Also, representatives from RADventure will be providing a demonstration of their Fenix ASP.NET Generator. If all this is not incentive enough to attend, take a good long look at the list of products offered by the conference sponsors (<http://etc.kcug.org/sponsors/sponsors.asp>).

Posted Wednesday, April 14, 2004

[Castle Computer Moving Sale](#)

Castle Computer Technologies is having a sale on the ProcedureNotes Template (normally \$29, now \$21) and on the Wallpaper Template (normally \$15, now \$10).

Posted Wednesday, April 14, 2004

[Yet Another Clarion Wallpaper](#)

Jesus Moreno has made another Clarion wallpaper available.

Posted Wednesday, April 14, 2004

[SQL Script Site](#)

Vince Du Beau points out this site dedicated to SQL scripts.

Posted Wednesday, April 14, 2004

[#cw-talk Clarion IRC Chat](#)

Want to chat with other Clarion developers online? Ole-Morten Heien is currently hosting the Clarion IRC server at www.hdsoftware.no port 6667. To participate: Go to www.mirc.com and download and install the latest mIRC software. After installing this software, and filling out what ever you need to fill, make sure you don't have any window

opened in mIRC. Then press the button with the Folder and a Hammer. This takes you to the mIRC Options. Now, select All in IRC Network, push the Add button, and use the following settings: IRC Server 217.8.158.243 or www.hdsoftware.no, Port(s): 6667 (default). Press the Add button and connect to the server. Now you will get online and you can join the #CW-Talk channel by typing /join #cw-talk.

Posted Wednesday, April 14, 2004

The Fourth, And Last, ETC

Lee White has announced that ETC 2004 will be the last ETC for the foreseeable future. So if you've been putting off the decision to go, thinking you'll attend next time, you'll want to think again. The ETC conferences have set the standard for regional Clarion get-togethers; if you've been to one, you know what we mean; if you haven't, you ought to see it for yourself.

Posted Wednesday, April 14, 2004

etc2004 SQL Topic Poll

Shawn Mason, the presenter on SQL, has requested a poll of possible subjects for his presentation at etc2004. If you are going to attend the conference please take a few seconds to select the subject you would prefer. Also, if you are going to attend and have not registered yet, please check the box below the topic list. The checkbox is so Lee can get a better idea of the head count for the conference. If you've already registered, please do NOT check the box. Please, only vote once and only vote if you ARE coming to the conference.

Posted Thursday, April 08, 2004

ThinkData Server Migration

The ThinkData site is undergoing some changes due to a server migration and will be down for a few days. If you have any questions or want to purchase xmlFUSE, OutlookFUSE, qbFUSE, faxFUSE or actFUSE please contact ThinkData directly via e-mail at sales@thinkdata.com

Posted Thursday, April 08, 2004

BST 3.0 Released

Big Scheduler Tamer Suite, Version 3.0 is now available. C55/C6 ABC and Clarion Chain, ODBC/SQL support, added ODBC/SQL capabilities to all templates. Firebird C6 example app included. Modest cost increase and upgrade path.

Posted Thursday, April 08, 2004

UK Clarion News

The main topic for the next UK Clarion user group meeting is COM in Clarion. "Any version of Clarion can be used to gain full access to Microsoft COM Active Data Objects. No add-ons are necessary. There are actually two methods of achieving this: The Names Method, and the COM Object Pointers Method. In the former case no Microsoft COM knowledge is necessary. In the latter case some knowledge is necessary, but it can be used as a way of understanding how Microsoft COM works ... because it is perfectly possible to 'parallel' both Methods side-by-side. Veronica Chapman describes the basics of these methodologies with examples.

Posted Thursday, April 08, 2004

CPCS Support Schedule

Support for CPCS products will be unavailable from 4/8/2004 through 4/11/2004. All support issues will be handled as quickly as possible on 4/12/2004.

Posted Thursday, April 08, 2004

Bug Poster 2.0 Available

Bug Poster V2.0 (freeware) is now available.

Posted Thursday, April 08, 2004

cpTracker Pro 2.0 Now Available

The latest version of cpTracker is now available (version 2.0). This version does require a new registration as the registration product name has changed from cpTracker to cpTracker Pro. There is a \$20 upgrade fee for existing users. Be sure and backup your prior version and data before installing as many files will be converted. The file changes and field expansions are detailed in the release notes.

Posted Thursday, April 08, 2004

[solid software Holiday Schedule](#)

The solid software office will be closed from April 4 until April 12.

Posted Thursday, April 08, 2004

[DevCon 2004 Call for Papers](#)

If you're interested in sharing a technical paper, a technique, or a code example with the Clarion community, SoftVelocity wants to hear from you. You can also contribute by presenting a tutorial, or hosting a code walk-through. In addition to helping the Clarion community, you'll also get into DevCon for free. Suggested topics list is available. Note: submissions of a marketing nature are inappropriate for this event and will not be accepted.

Posted Thursday, April 08, 2004

[Buggy Version 4](#)

The bug tracking tool Buggy Version 4 is now available. New features include: Multi-level program tree; SMTP support; HTML EMail support; Import Wizard; Updated help files; New manuals in PDF format; and more.

Posted Thursday, April 08, 2004

[DevCon 2004 Highlights](#)

DevCon 2004 conference highlights are now available on the SV web site.

Posted Thursday, April 08, 2004

[xWhatsNew Class 1.4](#)

SealSoft's xWhatsNew Class v1.4 is now available. Changes include: Code to generate xWhatsNew file moved to the global extension; New XP manifest option in Global Extension template; New option in main extension to refuse creation of xWhatsNew item in the Frame menu; To call xWhatsNew window manually you can use a code template.

Posted Thursday, April 08, 2004

[ClarionPost Product Listings](#)

If you are selling or providing any tools for Clarion, you can list them at this site.

Posted Thursday, April 08, 2004

ClarionNET For Clarion 6

ClarionNET and the ClarionNET Deployment Manager for Clarion 6.0 are ready for download and testing. The ClarionShop ordering info will be updated as their time permits. This version will operate in demo mode until the upgrade is purchased. In demo mode an "Unlicensed Demo" message appears during client connections, but there is no limit on functionality. Also there will be just one product for sale: ClarionNET & ClarionNET Deployment Manager are now bundled together. This applies to new purchases and upgrades. The Deployment Manager Load Balancer module will now manage up to 100 servers per deployment at no extra cost. The additional purchase requirement based on number of managed servers is no longer used.

Posted Thursday, April 08, 2004

xRuntimeStyle Manager 1.6

SealSoft's xRuntimeStyle Manager v1.6 is now available. Changes include: New option for display style tooltip; Now you can enter valid Clarion text for evaluate it in runtime. Updated Documentation, demonstration program and installation kits for Clarion 5.5 and Clarion 6 are available.

Posted Thursday, April 08, 2004

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



A Hunka-Hunka Burnin' CD Data, or Two Steps Forward, One Step Back(up)

by Mark Riffey

Published 2004-04-15

A long time ago in a place far, far away (Missouri, 1998), I started nagging our users to start backing up their data. My company, [Granite Bear](#), develops and markets Photo One, a professional photo studio management application. The data stored in Photo One documents every aspect of a studio's business, therefore backups are critically important.

The ability to burn data to CD from a Clarion 6 program is what you will get out of this article. Before I get into the code necessary to accomplish that task, I need to describe the scenario that motivated my desire to simplify that task.

If you look around at photography and cultured marble industry publications, you may find that I have been a bit of a nagmeister about backups and the other things that we programmers do as a matter of course to protect ourselves from disaster.

Originally, the nagging started for purely selfish reasons. TPS database repair support calls were taking up more and more of our time as our user base grew. Nagging wasn't enough, however. Photo One clearly needed an integrated backup solution, and more recently, the ability to back up to CD.

In most cases, the support calls were from users who needed to resolve database problems after a catastrophic data loss. As you would expect, the failure to backup almost always occurs just prior to power losses on computers with no battery backup and/or hard drive crashes. Sometimes the failure to backup stretches for months or years. No, really....

Not long after I raised my nagging to a fever pitch, it became obvious that we at [Granite Bear](#) had to work past a few issues with users and backups before we were going to make serious headway into reducing our "Backup, Oh sure, I know I did one last year" issues.

Specifically, the issues we wanted to resolve were as follows:

1. Most users don't know Windows has a backup program in it.
2. Most users don't know that the built-in Windows backup program leaves a lot to be desired.
3. Most users won't run a backup unless doing so is as simple as banging their heads against the keyboard.
4. Most users tend to assume that a program always backs up the data files it uses, just like Microsoft Word

As a result of these DUH-class findings, I purchased [Lindersoft's](#) fine LSZIP product with the intent of creating my own backup program. As it happens, I ended up using the sample "backrest" program as the basis of our backup utility. A day or so later, I had a one-button backup program that would create a zip of our TPS databases. We called it BearBack.

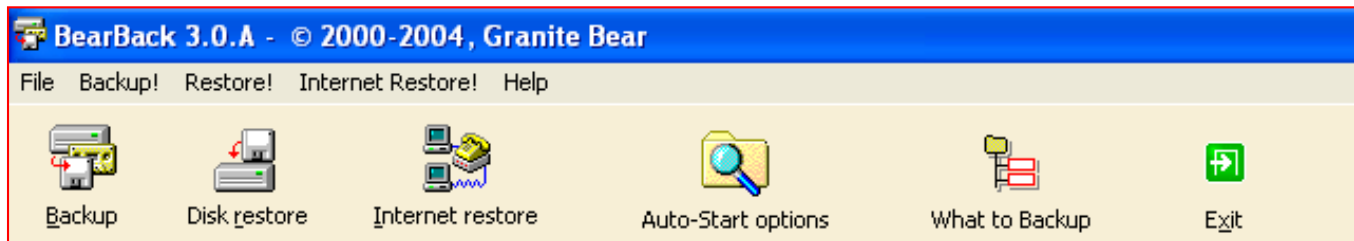


Figure 1. The BearBack toolbar

Users click the Backup button to start the backup process. The screen in Figure 2 is the next and final step before the backup process begins.

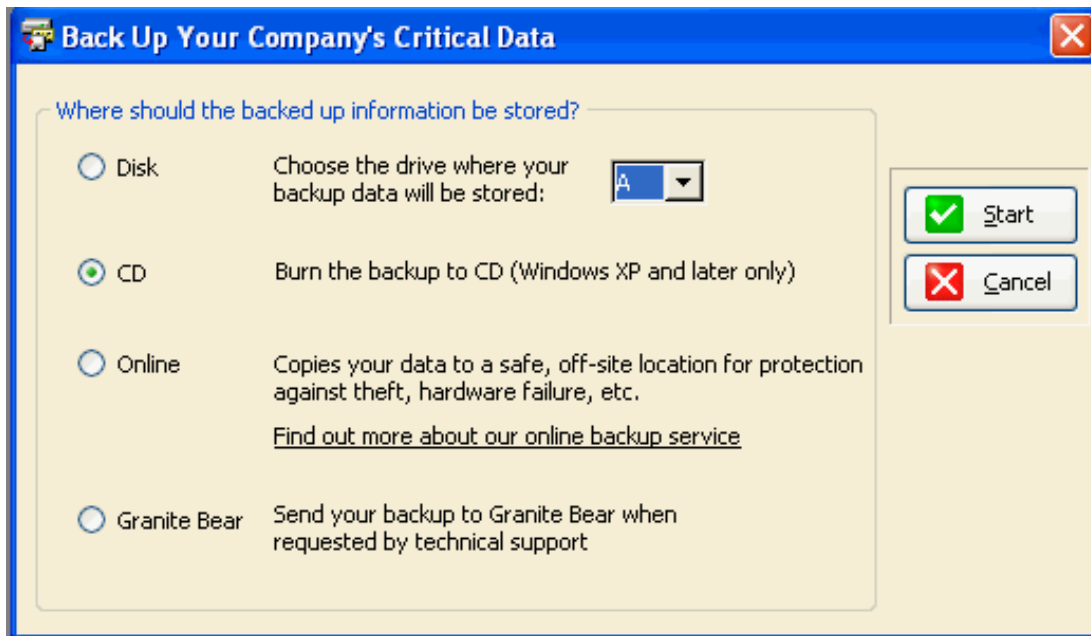


Figure 2: BearBack's "Where do I put the backup" screen

It's important to note that BearBack is not a world-class, all-encompassing backup utility. It was designed to be simple. It performs one task – it backs up Photo One data so that an end user or studio consultant can get a studio back on its feet using any commonly available zip tool, including BearBack itself. BearBack backs up all the .TPS, .DOT (Word templates we use with Arco Word Reporter), .XLS and image files. It does not look in other folders. It does not do incremental backups.

Those of you looking at the image in Figure 2 might wonder about the XP look and feel. This was accomplished using every ounce of artistic talent that I could muster, as well as a little bit of help from the following Clarion add-ons:

- [XP Manifest](#) from Power Office
- [Hyperactive](#) from Capesoft
- [Icons XP Group 1](#) from 1st Logo Design

Fast forward a few years. BearBack now backs up to and restores from any drive letter, uses FTP to back up and/or restore files using our off site backup service, uses FTP to send user files directly to our support team, and is capable of being run as an unattended task in any of the previously mentioned backup modes.

Still, this is not enough. Users want to back up to CD, even if that CD doesn't leave the CD drive for months (surely you don't think I'm kidding). And while most computers these days have DirectCD or similar software, which lets you treat a CD like any other drive, CDs simply aren't all that dependable when used as a direct target of the zip file creation process. We had not yet gotten around to putting CD burn capabilities into BearBack, but we knew we would have to do so in the near future.

It is obvious that many of our users are capable of using Roxio and a number of other programs to burn a CD, and of course, a number of them do exactly that. What isn't so obvious is that the majority of our users don't want to use anyone else's software if they can avoid it. They easily overwhelmed by the plethora of options in Roxio, Nero etc. They not only enjoy, but often demand, the ability to use one vendor's software to take care of their studio because it eliminates complexity, finger pointing and similar issues. Naturally, those who cannot burn their own CD backups are the ones most likely to experience catastrophic hardware failures. They are also the ones most likely to backup once a quarter, and/or backup the wrong files.

1.21 gigawatts and a flux capacitor later, we have fast forwarded ourselves to the near future. I am in the process of writing a digital photography presentation sales program to accompany our photography studio management software. Part of every digital photographer's workflow includes backing up their original image files to CD prior to any type of manipulation. As a result, I need to create a function in our sales program to handle this requirement.

In most cases, a zip file would be a suitable backup, thanks to the inherent integrity checking of the zip format. However, a zip file isn't good enough when you could be working with 100 to 200 megabytes of image files per photo session. In an incredible moment of clarity, I decided that burning a CD would be best. While I could create a backup folder and tell the user to burn a CD, that's a bit more difficult than the backup task I am looking for.

My quest: To find a component or source code that will allow my code to write a CD.

Google found very little, aside from a component that was very expensive and not really what I had in mind.

My next stop was SourceForge.net. I quickly found a nice CD writing project, but there were simply too many red flags popping up after I read details of the project and its capabilities. Too bad.

Next stop, [MSDN](#). I happened to stumble across some neat .NET code to talk to the [ICDBurn](#) COM interface. It's a nifty interface with a lot of control over the CD drive, but I didn't need to spend the next three weeks writing and testing CD burning COM code, so I filed that under "later, maybe" and moved on.

The Windows XP CD burn wizard

After resuming my MSDN search, I stumbled across a discussion of how to customize the Windows XP CD burn wizard. I found out more than I really wanted to know about the wizard, but more importantly, I found a nice bonus that fit our users' needs perfectly.

The bonus? I found the default special folder name for the XP CD burn preparation area and the API call used to locate it.

This XP CD burn area is a hidden folder that is normally located deep inside the logged-in user's documents and settings folder, but users can change the location via the CD device properties screen. Obviously, it's not safe to make assumptions about the location of this area. Back to MSDN I went. As it turns out, the API call used to find the location is `SHGetFolderPath`. This code should be placed inside your global map:

```
!SHGetFolderPath
MODULE('shell32')
    SHGetFolderPath(Long,Long,Long,Long,*CSTRING),|
        Long,PASCAL,RAW,NAME('SHGetFolderPathA')
END
```

In order to be able to link the program that calls `SHGetFolderPath`, you'll need to make a

lib file for shell32.dll (available at the end of this article).

The following code should be placed somewhere in your declarations. I put mine in the global area.

```
CSIDL_CDBURN_AREA equate(3BH) ! cd burning area for XP
SHGFP_TYPE_CURRENT EQUATE(0) !current value for user
```

This code can be a local variable:

```
loc:CDBurnPath cstring(1001)
```

Ok, now I'm ready to burn a CD.

Because the CD burn wizard feature is limited to Windows XP and subsequent releases of Windows, I decided to make sure the program was running on a compatible release (XP, Windows 2003) when the user attempted a backup. There's just nothing as fun taking a support call from an outraged user who is trying to burn a CD under Windows 95 on a computer with no CD burner.

In Clarion 6, I use `system{prop:WindowsVersion}` to find out if my program is running on Windows XP or Windows 2003. If so, I issue the `SHGetFolderPath` and then copy my backup file to the CD burn prep area.

Here's the rest of the code that makes the CD burn happen. I should note that prior to running this code, I have already backed up the user's data using `LSZIP` and my `backup.zip` is lounging comfortably on the C: drive.

```
if instring('WINDOWS XP',|
upper(clip(SYSTEM{PROP:WindowsVersion})),1)|
or instring('2003',upper(clip(SYSTEM{PROP:WindowsVersion})),1)
  IF SHGetFolderPath(0{prop:handle}, CSIDL_CDBURN_AREA, |
  0, SHGFP_TYPE_CURRENT,loc:CDBurnpath) = 0
    copy ('C:\backup.zip',clip(loc:CDBurnPath) |
    & '\MyCDBackup.zip')
    setcursor()
    message ('Your backup has been sent to the Windows ' |
    & 'CD burn prep area". In just a moment, Windows ' |
    & 'will inform you that a CD is ready to burn',|
    'Cmag',icon:asterisk)
  else
    message ('Sorry, this program cannot use the XP ' |
    & '"CD burn feature" because this computer ' |
    either has no CD writer, or your user account does ' |
    & 'not have Windows security rights to burn a ' |
    & 'CD, or the Windows CD writer properties ' |
    & 'area does not include a "CD burn folder".|| ' |
    & 'Please check each of these items before ' |
    & 'attempting to burn another CD. Your backup is ' |
    & 'complete and intact, despite the CD write failure.'|
    , 'Cmag',icon:hand)
```

```
display  
END!if  
end!if
```

Once this code completes, Windows will pop up a small window adjacent to the system tray, notifying me that there is data ready to burn to a CD.

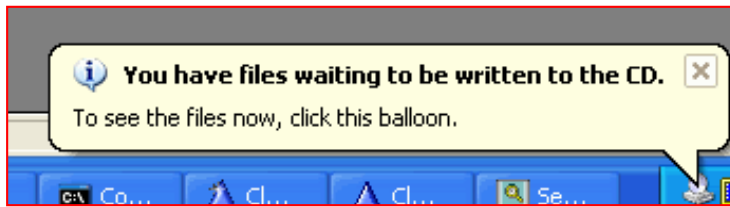


Figure 3. The Windows XP CD burn notification window

When I click the "You have files waiting to be written to the CD" popup message in Figure 3, Windows displays the screen in Figure 4.

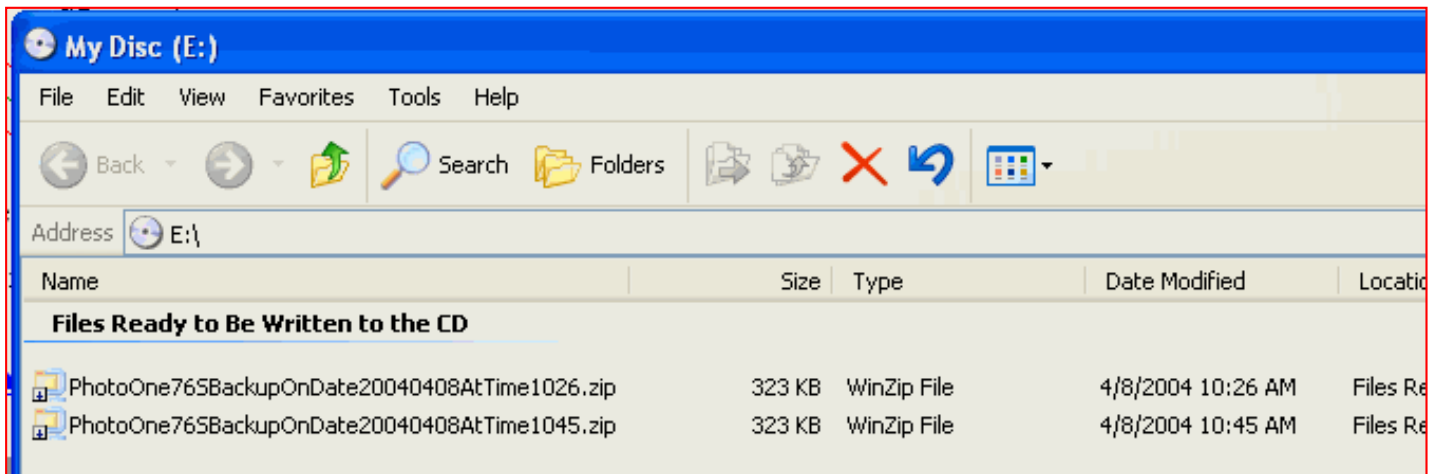


Figure 4. The Window XP CD burn wizard

Note the file name of the backup includes the date, time and Photo One version. We use this naming convention primarily to help our technical support staff resolve issues more quickly when "Which backup is the current one?" becomes an important question.

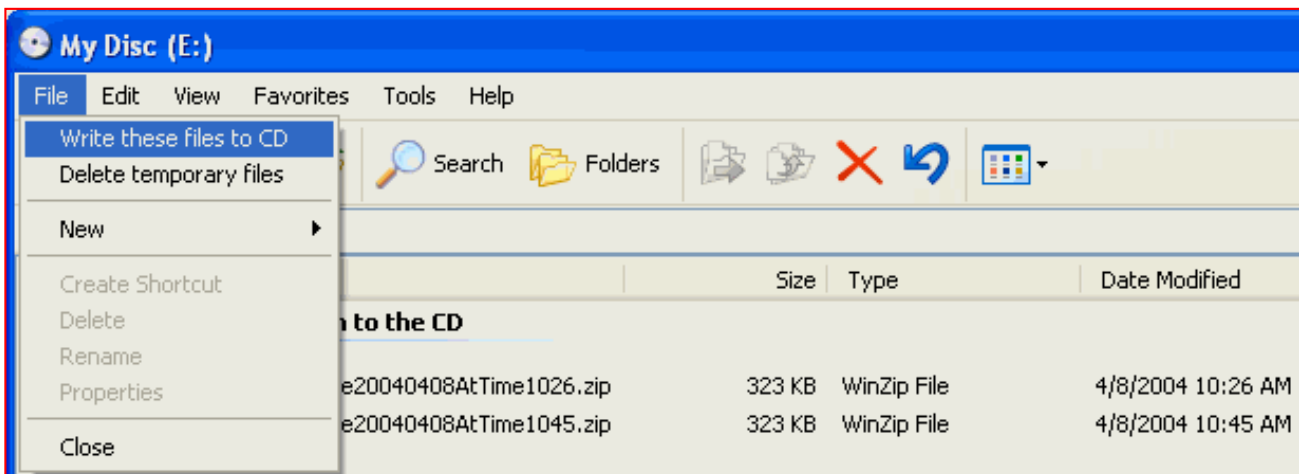


Figure 5: Writing the CD

While I hesitate to burn anything after dealing with Montana's forest fires over the last three years, I have found this code very useful for letting our customers create backup CDs using Windows XP's CD burn wizard.

[Download the source](#)

Mark Riffey has worked in the software industry, primarily in development and technical support for two internationally known enterprise software vendors, the world's premier information systems services company, a Fortune 100 manufacturer, and now Granite Bear Development. His business philosophy is simple: Be fair to your customers and yourself, surround yourself with brilliant people, work hard, be a good listener and have a little fun. Mark and his wife Jacki have two boys, Alex and Jonathan. Mark's other interests include Boy/Cub Scouting, backpacking/hiking and almost anything else outdoors, classic blues guitar, golf and photography.

Reader Comments

[Add a comment](#)

Mark Thanks for the article which is very...

Hi Barry, We no longer support 9x/ME for a number of...

[Clarion Magazine](#)



The Shortest Useful Clarion Program

by Tim Phillips

Published 2004-04-16

How little code can you write and still have a broadly useful program?

I try to write as little code as I can. I trust templates to carry the bulk of the burden instead of risking the logical errors and mistakes of hand code. Still, at times I have a need for hand code and - accidentally - I sometimes stretch the boundaries and discover a new limit in being efficient with my code. In this article I'll show you a small utility that demonstrates how much you can accomplish with just a little code.

A few years ago I found myself very annoyed at how hard it was to obtain a complete path and file name from Windows Explorer. Using pen and paper and writing things down longhand to transpose later was not - and is not - my style. Frustrated, I dug into the Clarion language guide searching for components to eliminate the problem.

What I ended up with was a utility program with a sum total of two lines of code that I wrote personally. I create a shortcut to this program in my Documents (I'll explain the SendTo folder later in this article). That way I can highlight a folder or filename in Windows Explorer, right-click on it and use the Send To options on the context menu to push the folder/file name to that application. The application then puts that folder/file name in into the windows clipboard so I can paste it into a document or tool that I am working with.

The application is small, but it demonstrates how to use the Windows clipboard, how to use runtime parameters in a program, and how to deal with the difference between long and short path names for files. It also demonstrates one of the few occasions I know of where Legacy code is superior to ABC and shows you the respective size differences of compiling a program as Legacy or ABC app, or as a hand coded project.

I built this applet (my term for a small utility program) by creating a new application without using a dictionary. The Main procedure is created as a source procedure and the following code is added to the Processed Code embed.

```
SetClipboard(CLIP(LongPath(Command('1'))))  
Message('Sent Name: ' & CLIP(LongPath(Command('1')))) |  
    & ' to the clipboard')
```

Let's review the language elements used and what they are doing

`Command()` returns command-line parameters present when the application is started. The '1' parameter will get back the first parameter on the command line. With the configuration I will use for a shortcut to call the applet, the returned value of `Command('1')` will be the file path name that I want on the clipboard.

`LongPath()` converts a short file path to a long file path. This extra step is needed when using this program on Windows 95/98 machines as they send the short file name along when the `SendTo` options are used (and long file names are so much easier to read and understand than the short format).

`CLIP()` cuts any extra spaces from the end of the file name so what gets into the clipboard is as short as possible.

`SetClipboard()` causes the contents of the Windows clipboard to be replaced by whatever is passed into the function.

`Message()` is the standard Clarion command for displaying a small

chunk of text to a user on a window. In this case, the user will see "Sent Name: <name of file> to the clipboard" with an OK button to close the window.

These two lines of code, plus the SendTo shortcut, will load the file name via a right mouse click into the clipboard and then display a message window that will permit a user to confirm that they didn't wiggle their hand at the wrong moment and have in fact copied the proper file name into the clipboard.

Although I used an application, these two lines of code can be compiled into a working program in three different ways: via an ABC source code procedure, via a Legacy source code procedure or via a Project with a CLW file.

When compiled as an ABC source code procedure, the EXE is 766K with the RunTime Library set to Local (meaning all necessary libraries are included).

When compiled as a Legacy source code procedure, the EXE is 468K with the RunTime Library set to Local.

When compiled as a Project, the EXE is 231K with the RunTime Library set to Local.

Creating a PRJ version

I'll leave the ABC and Legacy options as an exercise for the reader. Here are the instructions for creating this applet as a PRJ.

Choose File|New|Project. Type in a name like shortest.prj, and press OK.

In the New Project dialog set the main file to shortest.clw. Press tab (to fill in the EXE name). Click OK.

Now create a file called shortest.clw and paste the following code into

it:

```
program
```

```
map  
end
```

```
code  
SetClipboard(CLIP(LongPath(Command('1'))))  
Message('Sent Name: ' & CLIP(LongPath(Command('1')))) |  
  & ' to the clipboard')
```

(Special thanks to Dave Harms for the project example.)

As the size comparison shows, the most compact version of this program is created via compiling as a Project. The next trimmest is in Legacy code. The bulkiest is the ABC app.

You can see the reasons for this by just examining the source code (the CLW files) that get created by each approach. Even a "simple" Legacy program includes a pile of "extra stuff". Not as bad as the class libraries that are included with the ABC method, but more code than the six lines in shortest.clw as used by the Project method.

If you're creating this program as an APP instead of a PRJ, the difference in space is the only benefit I've ever found for continuing to use Legacy code. In this single instance, there is no need for the greater power of the ABC library.

In truth, with the vast size of modern hard drives I built the production version of this application as an ABC source procedure, since I use ABC for literally everything, and maintaining that consistency is more important to me than saving such a small volume of space.

The last step to using this little applet is defining a shortcut to activate it.

On a Windows 95/98 machine, you need to add a shortcut to

C:\windows\SendTo. On Windows NT4, 2000 Professional, XP Professional you need to add the shortcut to C:\Documents and Settings\\SendTo. (The SendTo folder is normally hidden, so you may have to configure Windows Explorer to show hidden files before you can access the folder to create a shortcut.) The shortcut just points to the exe, and has a descriptive name (FileName To Clipboard works just fine for me).

Once this shortcut is added, it should be possible to go to any file or folder on the system, highlight it, right-mouse to get the property window, select Send To, select your new shortcut and push the folder or filename into the clipboard.

There you have it: two lines of hand code, knowing where to put a shortcut so Windows can find it, and you have a feature that Windows Explorer should have built right into accessible and ready to go.

It's perhaps the greatest return for the smallest amount of hand code that I have ever seen.

Editor's note: I'd like to see more examples of really useful Clarion programs that take just a few lines of code. If you have something to contribute, post it as a reader comment, or if you think it has the makings of an article, [email it](#) to me. dh

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

Reader Comments

[Add a comment](#)

Excellent I've just implemented it though I managed to...
I love the DOS command prompt. A tip is that you can...
On Send To I frequently add Notepad.exe so I can right...
Carl, Nice tip. I always add notepad to my SendTo...
Carl's example can be downloaded at...

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Book Review: SQL Tuning

by David Harms

Published 2004-04-16



[SQL Tuning](#)

By Dan Tow

O'Reilly, November 2003

ISBN: 0-596-00573-3

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

The last time I [polled](#) ClarionMag readers about how much of their time was spent developing for SQL databases, over half said 25% or more, and 18% said they developed only for SQL. Now, developing applications that use SQL databases is one thing – making sure those applications make efficient use of the SQL server is another.

Like many developers who use SQL, I started off simply by creating a Clarion application that used an SQL database (MySQL, in my case). I let the ABC templates/classes create and manage the underlying views, which through the ODBC driver created the SQL statements. But soon enough I wanted to do some things with the data that ABC didn't know how to do. So I started writing SQL statements, and executing them with PROP:SQL, or via the MySQL client program. I

soon had a library of SQL statements, some of which worked well, and others which took horrendously long to complete. I'm not a DBA, but clearly I'd learned enough about SQL to become dangerous. Eventually, by trial and error (and by upgrading to a more recent version of MySQL), I solved most of those problems, or at least made the execution of those queries tolerable. But I can't say I really learned how to optimize my SQL statements.

Optimizing SQL is really what this book is about. It is *not* about how to optimize an SQL server. That is, it won't tell you what settings the server needs to run any given SQL statements as quickly as possible; instead it provides a methodology for optimizing those statements so they load the server as little as possible. This is an important distinction – server tuning is important, but it's no substitute for SQL statement tuning. And although the book focuses on DB2, Oracle, and SQL Server, the basic concepts, as well as the methodology, described here can be applied to SQL servers in general.

The author, Dan Tow, describes three key aspects of SQL tuning. The first, is to find and interpret the *execution plan* of the SQL statement. This is information, which the server will provide on request, explaining how it processes a particular query. The execution plan includes information such as which tables are processed in which order, and which keys (if any) are used. The second is to modify the SQL statement so that it uses the best possible execution plan. The third, and most important, aspect is knowing how to decide which is the best execution plan. Many developers make this decision intuitively, or by trial and error, using best guesses; Tow, however, provides a methodology, complete with diagramming technique, for finding the best execution plan.

This is by no means a beginner SQL book, but Tow does spend a chapter on data access basics, from a behind-the-scenes perspective. When he explains tables, for instance, he points out the importance of knowing how the physical layout of the tables affects read performance. When he discusses indexes, he explains how B-trees work, and looks at more exotic solutions such as index-organized

tables and bitmapped indexes. And of particular interest, he describes how SQL servers go about processing JOIN statements.

This chapter is followed by the requisite two chapters covering the topics of obtaining and understanding execution plans (with full examples for each of the three subject databases), and of modifying SQL statements to achieve a different execution plan. These modifications can force the server to use or ignore specific indexes, use or prevent join orders, and control the order in which outer queries and subqueries are executed. Again, there are specific examples for DB2, Oracle, and SQL Server.

Most of the rest of the book deals with Tow's tuning methodology, which uses a diagramming notation that focuses specifically on those parts of a query which really do affect performance, including the proportion of records returned from any table in a join (the *filter ratio*), and average number of records found on each side of the join (the *join ratio*). The diagram ignores those things which have little or no affect, or which cannot effectively be optimized anyway (columns selected, ORDER BY and GROUP BY clauses, table names, details of join conditions, absolute table size, etc.).

The diagramming notation itself is remarkably simple – the hardest part (which is not that hard) is getting the data on join ratios and filter ratios. Once you have that you can lay out the diagram, and once you've laid out the diagram you follow a few simple rules to deduce the best execution plan. Tow's examples range from simple two-table joins to monsters of 17 tables or more. And he does have experience with massive SQL statements – he writes that he routinely tunes joins with more than 40 tables, and his personal record is 115 tables.

The chapter on tuning complex statements (and no, a 17 table join isn't necessarily complex, at least according to Tow) is also a lesson in database design. When a SQL statement becomes too complex to diagram quickly, often the reason is a flaw in the database itself, and usually in the design of the application that uses the database.

Possible flaws include cyclic joins, two disconnected queries combined in a single query, query diagrams with multiple roots, joins with no primary key, and problematic one-to-one joins, among others. There is also a brief chapter on why the diagramming method works, and several final chapters on special cases and solutions to "unsolvable" problems.

As evidenced by other reviews I've read of this book, Dan Tow is clearly an individual with a keen understanding of how SQL servers work, and how they can be made to work most efficiently. His diagramming notation is simple and effective, and his methodology highly effective. If you've been using SQL and you're starting to bump up against performance issues, or if you'd like to make your SQL applications more robust, or if you're just curious about how SQL servers handle multi-table queries, then you need to read this book.

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

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Clarion Magazine](#)



Postmortems and The Clarion Tips & Techniques Book

by Andrew Guidroz II

Published 2004-04-22

A visit to a client site to make a few software changes promised to be uneventful and quiet. But I try to learn a little from every single day's work. This trip was no exception.

The client at the site I visited rates customers' creditworthiness using two different scores. These scores are computed by a Clarion application which reads through the customer's past history, which is actually data that is exported from a legacy computer system. The techniques of computing each score are similar and involved reading through the same transactions. But, because the two scores are needed in different procedures, I wrote the code as two different routines, one in each procedure.

Times are changing and now the legacy computer system has gone away – only the old data remains. The routines needed to be modified to check the legacy data first and then check the new and improved data from the new system. It sounded like two changes to two routines. I immediately thought it would be a good time to go ahead and combine the routines into a single procedure that would return both scores at the same time. The amount of overhead to return two instead of one was minimal and the client would be getting more information by having access to both scores.

I wrote the source procedure to compute the two scores and noticed that I was always getting zero back for both scores. I checked my prototype and saw it was defined as (`*LONG, *LONG`), using the old syntax of just specifying the data types in the parameter list. This allowed for calling by address so my new procedure could change the value of the two parameters I was passing. But something wasn't working, and I spent a while re-reading the help file on prototyping and parameter passing.

The clock was running and I needed to finish soon. I also wondered about the [newer styled parameter passing](#) (`*LONG parScore1, *LONG parScore2`), and really wanted to use it but I didn't have the time to remember what stumbling block kept me from using it in the past.

I re-read my code and the problem was readily apparent. I don't tend to change the values of parameters throughout my source code. Instead, I typically move them to local variables at the entry point, e.g. `Loc:NewScore1=ParScore1`, manipulate the local variables, and then, if they have to be returned with new values, make assignments to the parameter variable near the single exit point of my procedure. This is a form of self documentation in that I can explicitly see the `ParScore1 = Loc:NewScore1` just before the exit. I don't have to recheck the prototype to know that I'm trying to return a changed value.

Because I was in a rush, I typed `Loc:NewScore1 = Loc:NewScore1` at the end of my code, and so I never assigned a value to the parameter. I fixed the code and I was back on my way. I made the change to the routines in the calling procedures from running through dozens of lines of code to a single procedure call. The changes worked great.

It was time for one more "cover your rear end" move. I fired up [Clarion Source Search](#) from Carl Barnes and did a search for any occurrence of the legacy data file name in this application. That way, I could make sure to call this new procedure that read the new data rather than report incorrect information by reading only the old

legacy stuff. And Source Search would tell me if I had put the old code somewhere else.

Source Search returned an incredible number of hits, approaching 50 occurrences.

How could this be? How could I have copy and pasted similar code in 50 different places throughout my application, and never felt motivated to turn it into a function or procedure?

A junior programmer had been brought in over the last few months to produce some reports. He saw the routine, realized he needed it, and was busy copying and pasting it everywhere. No big problem for me. The routine was identical in every occurrence, so each routine now became a single procedure call. I had a successful day and left notes for the junior programmer to show him how to take advantage of this new procedure.

But I never leave a project like this alone. I always go back over in my head where the stumbling blocks were in my own work. Folks who work on large projects have what they call a *postmortem*. This is when you go over what went well in your project and what went wrong. It is always very educational and can aid you in your next project.

I take this concept a bit further and attempt to do a little of this with every day's work. I had a false start with prototyping and parameter passing, or with what I *thought* was a problem there. I made a mental note to re-read those sections in the Clarion documentation to make sure I truly understood it, and see if there was anything new in C6's documentation. But a new piece of documentation was sitting on my night stand.

I picked up [Clarion Tips and Techniques](#) and saw the article "[Procedure Prototypes](#)" by David Harms. This article showed the usefulness of the newer style of prototyping supported by Clarion 4 and later versions. It also helped to remind me why I had stayed

away from this in the past: the text box to prototype was rather small in C4 through C5.5. In Clarion 6, I can now pop up an entire window to see the entire value in that text box by pressing the F10 key. This is called the Zoom feature.

["The Nuts and Bolts of Passing Parameters"](#) by James Cooke was just a few pages further on. This article helped to hammer home again exactly how parameter passing works. And it contained an unexpected bonus. I always assumed I was limited to passing up to three string parameters to any threaded procedure. An editor's note in the article referred to a [workaround](#) for this limitation by Jeff Slarve.

Postmortems are useful and powerful in your education and development into a better programmer, and [Clarion Tips & Techniques](#) is definitely another tool to include in your postmortem toolkit.

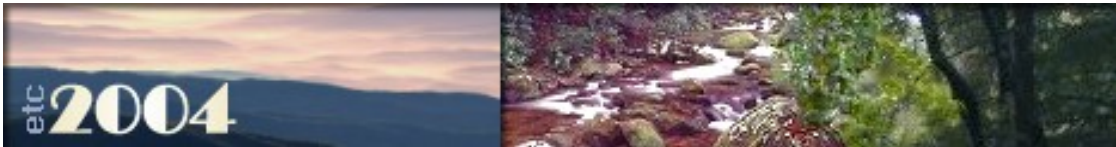
Andrew Guidroz II, when he isn't traveling around the countryside watching his 2004 National Champion LSU Fighting Tigers, writes software for all facets of the insurance and finance industries. His famous Cajun cookouts have become a central feature of Clarion conferences throughout the U.S. Andrew's Cajun website is www.coonass.com.

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Clarion Magazine](#)



Basic File Handling in Clarion

by Dermot Herron

Published 2004-04-23

From the Shorter Oxford English Dictionary:

Ritual. A series of actions compulsively performed under certain circumstances, the non-performance of which results in tension and anxiety.

This article is a very elementary introduction to basic hand-coded file handling using Clarion's ABC library. It is based on a talk I gave to the Johannesburg Clarion User Group in January, 2004. I'll be discussing my way of thinking about Clarion TPS files, and some standard approaches I've developed (rituals, if you like) for handling those files. This is certainly *not* the only way of doing this, but it is reliable, clear, obvious and very well tested over lots of years.

A mental image

To think about files in a computer I find I need a mental image of what is happening inside the computer. I like to think of a TPS (ISAM) file like the old library card catalog, or Cardex, for books. (See the pictures at the end.)

Imagine every *physical book* in the library has a corresponding 5x7 card with this information on it :-

BookTitle

| | |
|---------------|--|
| AuthorNo | (a unique number for each author) |
| AuthorName | (you would not duplicate this in a computer, but you did on a card for user convenience) |
| BookRefNo | (for each <i>physical</i> copy of the same book, a <i>different number</i> !) |
| Status | (if it is lent out etc.) |
| ReturnDate | (when borrowed, this is filled in) |
| Location | (where it is on the shelf and which shelf) |
| CopyrightDate | (when it was written) |
| ISBN-No. | (a unique number for that book title, binding etc.) |
| Other info | publisher, etc |

There will be several cards which are almost identical except for the BookRefNo for multiple copies of the same book and there may be many different books by the same author. These cards will be arranged in a drawer in Title order.

(In the old days, there was a primary Cardex arranging the cards in Title order and other *duplicated complete Cardexes* arranging them in author order and date-published order, etc. Imagine keeping this lot updated and accurate!)

Also imagine that there is a corresponding *author 5x7* card which contains :-

AuthorNo

AuthorName

<Other info about the author like a short life history>

Now to the computer...

In a TPS file think of KEYS as separate Cardexes with the records arranged in the key order. So cards arranged by title would have all the identically-titled books next to one another ("Where the Lion Feeds") , and if you arranged the cards by author, all the books by Wilbur Smith would be next to one-another. (There is no Cardex equivalent of books in RECORD order – one would not have wasted time creating a Cardex showing the order in which the library received the books!)

Imagine now that this is computerised and you use a file Books.tps, PRE(BKS) with keys on all the necessary fields. In the "Books" file I would expect keys on at least BookTitle, BookRefNo, AuthorNumber and ISBN-No.

Also imagine a file called Authors.tps, PRE(AUT) with keys on AuthorName and AuthorNo.

How the computer stores it

An ISAM database (like Topspeed's TPS files) stores the "cards" (= records) in captured order. The clever bit is that it also keeps physically separate lists (KEYS) which are sorted in a specified order (alphabetical, numeric, date, time etc). Each entry in the KEY then "points to" the actual record, or to use the analogy, a "card". This means you can look for books in any sequence you choose to set up. In reality, the computer actually makes, and keeps updated, another "Cardex" for each and every key, containing only the key component(s) and a pointer to the "real" card where the rest is stored. (With TPS files, the whole lot is stored in one file on disk, so you don't actually see the separate KEY "cardexes" but they are physically there.)

The Clarion *language* contains primitive file access (ADD, PUT, GET, DELETE, etc.) that allow the handling of records in a file. The ABC library adds memory and intelligence. When you declare a file in an application, you *automatically* get an instance of the ABC FileManager object. The syntax you use then is `Access:<NameOfFile>.<method>`, (e.g. `Access:Author.INSERT()`). The main methods used are INSERT (ADDs a record), UPDATE (PUTs a record), FETCH (GETs a record) and NEXT. Notice that

SET is the normal Clarion Language command and is *not* an ABC function – you use SET for both straight Clarion code and ABC.

Putting "cards" in is easy (`Access:Author.INSERT()`), and changing one card and putting it back is also easy (`Access:Author.UPDATE()`), but finding one or a series of cards with a specific characteristic is often the difficult bit.

The "Ritual" bit

After a lot of years programming (say, in quavering voice, "thirty-five") I have found methods that work for me, so I have developed "rituals" (often expressed as macros in the Clarion editor) with which I do a given thing exactly the same way every time. This means that I don't forget to do anything. Most of my rituals apply to file handling.

For example I actually call the `AuthorNo` "AutRef" in my ritual of using `<prefix>Ref` for the main reference key field in each file. I always name the keys `By<FieldName><FieldName>...` in the correct order. Since the prefix is guaranteed by Clarion to be unique in a given dictionary, and I know the prefix by frequent use, this makes it easy to remember. I use exactly the same field name in related files so the `PRE` of the file points to the file. I'm lazy so I keep my naming short but easy-to-remember. (I also use a lowercase `L` prefix for local variables and lowercase `G` for global variables, e.g. `lTimeTaken` and `gMainDate` – these are quicker to type than `loc:TimeTaken` or `glo:MainDate`.)

Finding records

There are two ways of retrieving records. Of course you have to know *what* you want, which means knowing something about a record, like the author name, that you are interested in.

The method you use depends on the type of entry you're looking for.

1) To find *one* specific record:

If someone phones to ask about a book they have borrowed and wants to

know the return-date, you could ask for the book number from the card in front of the book. There is, necessarily, only one physical book with that number.

In that case, you would use the following code:

```
BKS:BookNo = lBookNo          ! you typed the number into lBookNo
IF Access:Books.FETCH(BKS:ByBookNo) ~= Level:Benign
  MESSAGE('Please ask for the book number again -' |
    & ' this number is not in the system')
  CYCLE
END
```

This code will retrieve that book's record. The important thing about `FETCH` is that it *only* retrieves an *exact* match, so a partial or incorrect book number will always return `LEVEL:Notify`. (If `FETCH` fails it clears the record buffer - `TryFetch` does *not* clear the record buffer))

2) To find a SET of records with a common factor (field(s))

Now let us say that you then want to retrieve all the books written by a given author after 1995, and list them in a report. Imagine that you have already looked up Wilbur Smith in the `AUTHOR "Cardex"` and found his number. Retrieving all his books requires multiple disk access, so you use the following code:

`lBookCount = 0 ! to count the books you have of his`

```
lBookCount = 0          ! to count the books you have of his

Bks:AuthorNo = Aut:AuthorNo
SET(Bks:ByAuthorNo, Bks:ByAuthorNo)      ! SET( <key>, <key>)
LOOP
  CASE Access:Books.NEXT()
  OF LEVEL:Benign                      ! Keep this bit together
    IF Bks:AuthorNo <> Aut:AuthorNo    ! for documentation
      BREAK                          ! so you understand
    END                                ! what and how
  OF LEVEL:Notify                       ! you are processing.
    BREAK
  OF LEVEL:Fatal                         ! This is a "ritual"
    STOP('Access:Books.NEXT()')
    BREAK
  END
```

```

! now check for other exclusions
IF BKS:CopyrightDate < lFromDate
  CYCLE
END
! etc...

! Do other exclusions here -
! don't get clever: just do
! only one thing in an IF line -
! it is self-explanatory
! like this.

! Finally we can process
lBookCount += 1
PRINT(?BookDetail)

! After all the exclusions,
! here we do the actual work...

END

! If there were any books then print the number
IF lBookCount
  PRINT(?NoOfBooksDetail)
END

```

Explanations:

Note that in both cases, you have to put a value into the key field *before* you initiate any search/retrieve. The computer is not intuitive about what you want.

SET(<key>, <key>) sets both the processing sequence to BKS:AuthorNo and the starting-point. NEXT then finds the *best* match for the value in the key field. This is the important difference – NEXT will do the best it can and return a near-match after or at the value in the key field. You *usually* get a record from NEXT, even if it is not the one you want, hence this piece of code restricts the result to exact matches:

```
IF Bks:AuthorNo <> Aut:AuthorNo THEN BREAK.
```

Contrast this with FETCH, where you don't get a record at all if there isn't a perfect match with the key.

If you use FETCH where there are multiple identical records, like looking for all the books written by an author, you will always only get the first book in the file. Subsequent FETCHs will *not* find the subsequent records.

I usually put in a complete CASE Access:Author.NEXT() and all three possible results, because if I have forgotten to open a file or some other silly error, it immediately comes up and I don't spend days looking for why the loop terminated prematurely (been there, done that...), which you would get

if you used the following much simpler code:

```
LOOP UNTIL Access:Authors.NEXT()  
    ! file processing  
END
```

Of course, you could use the `TryNext` method but I like Clarion to tell the user in detail what the error is, and I put in a `STOP` to make it very clear to the user that they must not continue.

Glossary

ISAM Indexed Sequential Access Method (e.g. Topspeed) This is a way of storing data on a disk – there are many different ways but this is one of the most-used and well-studied.

Declaring files in an application: You can cause Clarion to include a file (and to make an instance of `FileManager` for that file) by including the file in one of two places (it doesn't matter if you do both, but you lose some minor error-checking):

1. By putting the file into the `FILES` of any procedure
2. Tick the file in `Global | IndividualFileOverrides | "GenerateFileDeclaration"`

The real cardex

The following pictures are from a British library which is computerizing its cards. There are many rows of these drawers as you can see from the size of each letter.



Each card has a single hole punched in the bottom and they are held in the drawer by a long rod that screwed into the back of the drawer and goes through all the cards. This allows you to read them but not to remove them, and they don't fall out if you drop the drawer !



Reader Comments

[Add a comment](#)

**This and other talks of Dermot at the user group are very...
I hope Dermot will publish his other talks. I wou;d...**

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Clarion Magazine](#)

online sales and delivery
for your applications & tools

Developer **PLUS**

Understanding Clarion Templates, Part 3: Introducing Code Templates

by **David Harms**

Published 2004-04-23

The Clarion template system is, at heart, a simple idea. If you want to generate code, based on varying conditions and requirements, you just create a template programming language that lets you generate Clarion language statements. In other words, you create an interactive program that generates a Clarion program.

Now, if you go to the Clarion template directory and examine any given TPL or TWP file, the idea that the template system is "simple" may not sound like a particularly factual statement. After all, as I explained in [Part 2](#) of this series, templates are source files that are a mix of template language and Clarion language statements, and it can be difficult to disentangle the two.

It's true that the templates that ship with Clarion are, for the most part, complex and difficult to understand. But complexity is no more a requirement of the template language than it is of the Clarion language. You can write simple, easily used/understood Clarion code, and you do the same with templates. And one of the easiest ways to get started writing Clarion templates is with code templates.

The CloseCurrentWindow code template

Code templates are simply templates which you can insert in a single embed point. The templates generate code only for that embed point, and they may or

may not have prompts.

One of the simplest code templates is the `CloseCurrentWindow` template, which has shipped with Clarion since CW 1.0. To use this code template, you must first be in a list of embed points.

The full list of embeds for any procedure is available from the Embeds button on the procedure properties window. If you want to see just the embeds for a particular control, go to the window formatter, right click on the control, and choose Embeds.

Assuming you're in an embed list somewhere, select a likely embed point and click on the Insert button. The Select Embed Type window, shown in Figure 1, appears.

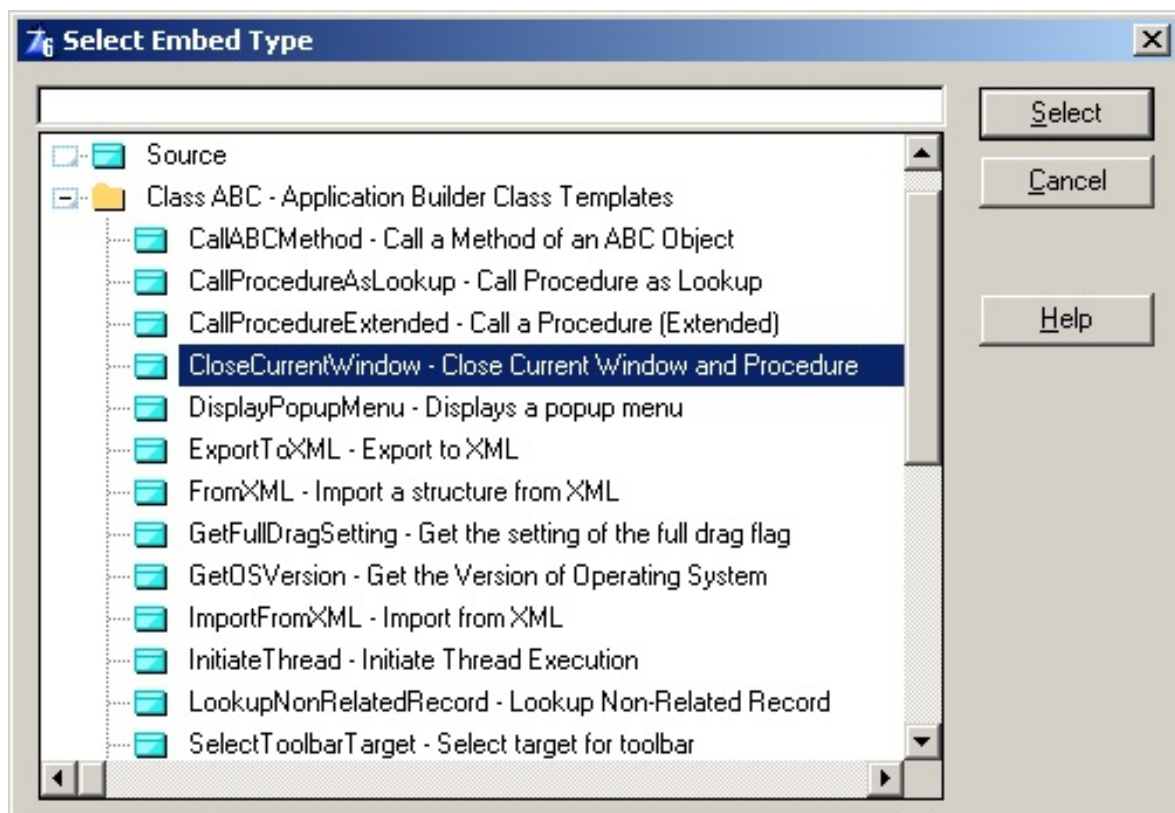


Figure 1. The Select Embed Type window

Highlight `CloseCurrentWindow` – Close Current Window and Procedure and click on Select. The dialog in Figure 2 appears.

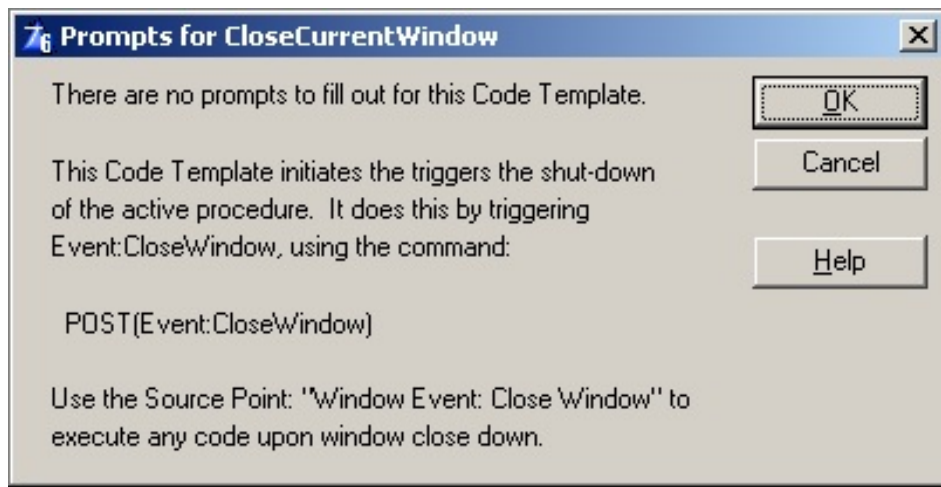


Figure 2. The CloseCurrentWindow code template information dialog

The window you see for CloseCurrentWindow is strictly informational. It tells you that there are no prompts to fill in (some code templates will have check boxes, entry controls, even file or field lists to choose from) and it gives you some information about the purpose of the template. Because this particular template contains only one line of code, that code is also displayed. Other templates may describe only the functionality. The point is simple—if the code template requires something of you, it will prompt you for what it needs in this window.

If you select OK, this code template will be hooked into the embed point you have selected. When you next do a Make or Generate All, or if you use the embeditor, the template will insert its code into the embed point.

You might wonder, in this case, why bother with a template at all. Wouldn't it be simpler just to write that one line of code? It might be, if you were quite familiar with the language. Even then, however, you would lose one important advantage the template system provides. If, at a later date, the command to close a window changes, you would need to go through all your source embed points and change `POST(EVENT:CloseWindow)` to whatever the new command might be. If you are using the template system, you would need to change the code template only once, and the next time you generated code all embed points that used that code template would reflect the change.

Here's the code for the template:

```
#CODE(CloseCurrentWindow,'Close Current Window and
Procedure'),HLP('~TPLCodeCloseCurrentWindow')
#DISPLAY('There are no prompts to fill out for this Code Template.')
```

```

#DISPLAY(' ')
#DISPLAY('This Code Template initiates the triggers the shut-down')
#DISPLAY('of the active procedure. It does this by triggering')
#DISPLAY('Event:CloseWindow, using the command:')
#DISPLAY(' ')
#DISPLAY(' POST(Event:CloseWindow)')
#DISPLAY(' ')
#DISPLAY('Use the Source Point: "Window Event: Close Window" to ')
#DISPLAY('execute any code upon window close down.')
    POST(Event:CloseWindow)

```

There are only three template language statements used in this template. The `#CODE` statement indicates this is a code template; the first parameter is the label which the AppGen will identify the template, and the second parameter is the description which appears on the template list. There is also a help attribute.

The `#DISPLAY` statements simply put text on the template's built-in display window. Although `#DISPLAY` can have a variety of attributes, including `AT` to specify positioning, this template has simple requirements and so the minimalist form of `#DISPLAY` is used.

Finally, there is one line of Clarion code:

```
POST(Event:CloseWindow)
```

This is the only part of the template that will appear in the generated source code.

The InitiateThread code template

Another code template that has been around since CW 1.0 is the `InitiateThread` template, which you can use to start a new procedure on its own thread. Figure 3 shows the prompts for this template.

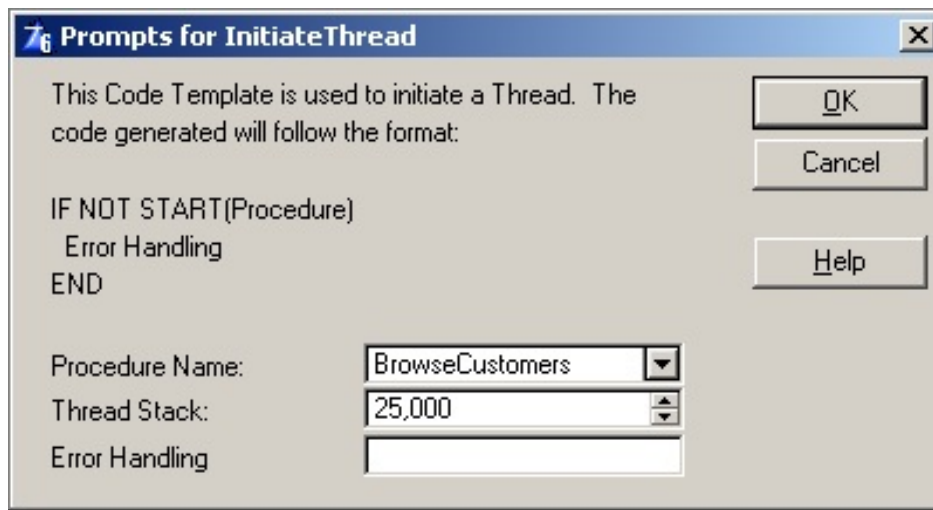


Figure 3. The InitiateThread code template prompts

This template is a little more involved than the `CloseCurrentWindow` template. Like that template, `InitiateThread`'s prompts give you an idea of what code the template will generate. The difference in this window from the previous is that there are prompts for you to fill in. You supply the procedure to call (you can type it in or pick one of the existing AppGen procedures), an optional stack size parameter, as well as some code to execute if the thread cannot be started.

The code for this template is as follows:

```
#CODE(InitiateThread,'Initiate Thread Execution'),HLP('~TPLCodeInitiateThread')
#DISPLAY('This Code Template is used to initiate a Thread. The')
#DISPLAY('code generated will follow the format:')
#DISPLAY('')
#DISPLAY('IF NOT START(Procedure)')
#DISPLAY(' Error Handling')
#DISPLAY('END')
#DISPLAY('')
#PROMPT('Procedure Name:',PROCEDURE),%ThreadProcedure,PROP(PROP:DropWidth,140)
#PROMPT('Thread Stack:',SPIN(@n7,5000,64000,500)),%ThreadStack,DEFAULT(25000)
#PROMPT('Error Handling',@S255),%ThreadError
#IF(%ThreadError)                                #! IF Thread Error provided
  #IF(%ThreadStack)
  IF NOT START(%ThreadProcedure,%ThreadStack) #&! IF Thread not initiated
  #ELSE
  IF NOT START(%ThreadProcedure)              #&! IF Thread not initiated
  #ENDIF
  %ThreadError                                #&! Perform the error code
END                                             #&! END (IF Thread not...)
#ELSE                                           #! ELSE (IF NOT Thread Error...)
  #IF(%ThreadStack)
  START(%ThreadProcedure,%ThreadStack)        #&! Initiate the Thread
  #ELSE
```

```
START(%ThreadProcedure)           #&! Initiate the Thread
  #ENDIF
#ENDIF                             #! END (IF Thread Error...)
```

Like CloseCurrentWindow, InitiateThread begins with a #CODE statement, followed by a number of #DISPLAY statements. The first difference is in the use of #PROMPT statements. These let the template collect information from the developer, and really it's mainly once you start using #PROMPTS that templates start getting a little complex, because now the template has to generate different kinds of code, based on what settings the developer has chosen.

The tradeoff is that there's a little more work to do up front to make sure that the template always generates correct code based on the supplied options, but once you've done that, you can simply forget about how the template works, and use it wherever and whenever necessary. Code templates are code fragments. In other words, they are made up of various portions of code that otherwise could not stand on their own and compile successfully. The rule of thumb is don't code anything more than twice—once to make sure it works, and the second time to place it in an appropriate template.

The first #PROMPT statement has a PROCEDURE attribute, which means that the IDE will supply a drop list of all the procedures available in the currently loaded application. If you like, you can also type in the name of a procedure without using the list.

The second #PROMPT has the SPIN attribute, which lets the developer choose a stack value for the START statement, with a value somewhere between 5000 and 64000 bytes.

The third #PROMPT lets you type some code to execute if START fails. The expectation is that you will enter a single line of code, perhaps something like MESSAGE('Unable to start this procedure'). You can enter more text than will display in the entry field. In Clarion 6, there is a zoom feature which lets you edit the field in a text box. Put the cursor in the Error Handling field and press F10. You'll see a text entry field as shown in Figure 4.

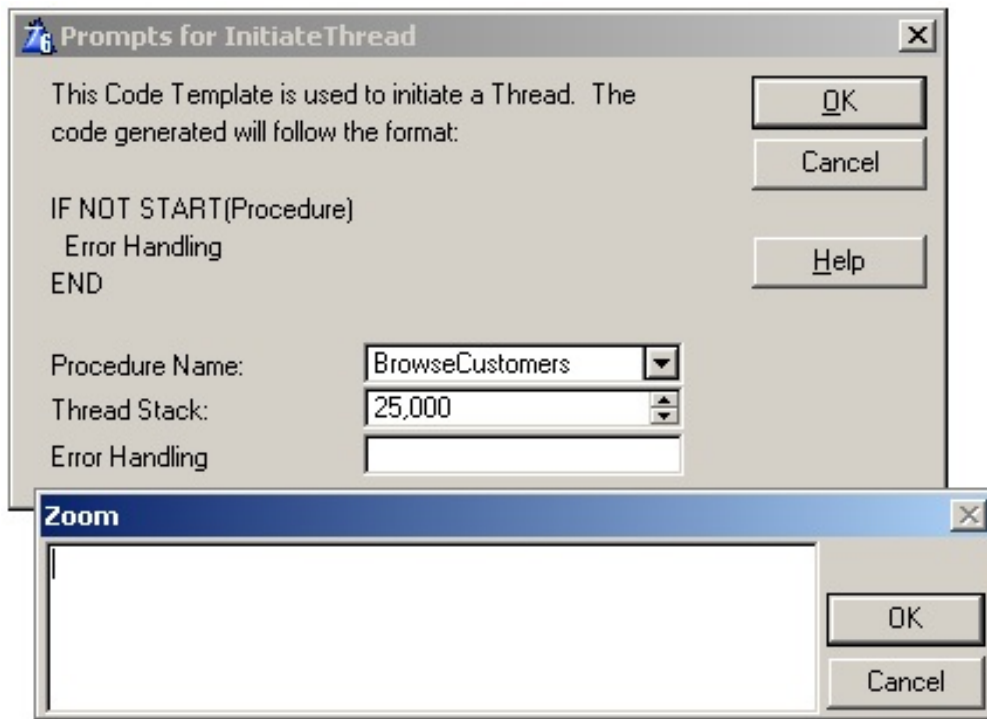


Figure 4. Zooming an entry field

NOTE: You can actually create multiple lines of code in this supposedly single line entry field. Simply press Ctrl-Enter or Shift-Enter to start a new line. But be warned! Only the first line will be automatically indented – subsequent lines will begin in column one, and you will get a compiler error unless you add a space at the beginning of the line, because the compiler will think you're writing a variable declaration, not code.

Generating the code

As this template has a number of options, the code it generates can vary. The first test the template makes is to see if the developer has entered any error handling code. If so, it generates the `START` code inside an `IF` test. As well, there are two forms of `START`, one with a stack size parameter, and one without. If omitted, the stack size defaults to 10,000 bytes.

```
#IF(%ThreadError)                                #! IF Thread Error provided
  #IF(%ThreadStack)
  IF NOT START(%ThreadProcedure,%ThreadStack)    #&! IF Thread not initiated
  #ELSE
  IF NOT START(%ThreadProcedure)                 #&! IF Thread not initiated
  #ENDIF
  %ThreadError                                    #&! Perform the error code
```



```
END                                     #&! END (IF Thread not...)
```

As it turns out, there's no way (at least that I found) to get the template to complete *without* supplying a stack size, so the remaining code will, I expect, never execute. In that case, the code part of the template could be shorted to:

```
#IF(%ThreadError)                       #! IF Thread Error provided
  IF NOT START(%ThreadProcedure,%ThreadStack) #&! IF Thread not initiated
    %ThreadError                             #&! Perform the error code
  END                                         #&! END (IF Thread not...)
#ELSE                                       #! ELSE (IF NOT Thread Error...)
  START(%ThreadProcedure,%ThreadStack)     #&! Initiate the Thread
#ENDIF                                       #! END (IF Thread Error...)
```

Another way to write this template would be as follows:

```
  IF NOT START(%ThreadProcedure,%ThreadStack) #&! IF Thread not initiated
#IF(%ThreadError)                       #! IF Thread Error provided
  %ThreadError                             #&! Perform the error code
#ENDIF                                       #! END (IF Thread Error...)
  END                                         #&! END (IF Thread not...)
```

or even

```
IF NOT START(%ThreadProcedure,%ThreadStack) #&! IF Thread not initiated
  %ThreadError                             #&! Perform the error code
END                                         #&! END (IF Thread not...)
```

Both alternatives have problems, however. If error code is supplied, everything looks normal:

```
IF NOT START(MyProc,25000)
  MESSAGE('Houston, we have a problem.')
END
```

If, however, the developer does not enter code in the Error Handling field, the first variation will generate this code:

```
IF NOT START(MyProc,25000)
END
```

and the second will generate this:

```
IF NOT START(MyProc,25000)
```

```
END
```

Neither of these conforms to the standard Clarion practice of using single line IF statements where possible, terminated by a period. In general, you find that where sacrifices are made in readability or style, they're more usually made in the templates, not in the generated code.

Now that you have the basic idea of what code templates can do, you're ready to move on to writing code templates. I'll begin that discussion next time.

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

Reader Comments

[Add a comment](#)

**Thank-you! Thank-you! Thank-you! I've been waiting for...
Tom, You're most welcome! Much more to...**

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Clarion Magazine](#)



Burning COM: How To Write CDs in Windows XP With ICDBurn

by Andrew Guidroz II

Published 2004-04-29

Nearly a year ago, I decided it was time for a career change. Over the last 20 years, I worked my way up the corporate ladder in the insurance industry and became a vice-president of a small insurance company, which also owned a number of other businesses. The company kept getting larger, and more and more of my time was taken up doing things like meeting with auditors, telephone companies, and even plumbers for dozens of locations. I was a long way from where I'd started, as a Cajun junior programmer. Computers had become infrastructure to me, just like plumbing, phones, and roofing, and I wasn't enjoying the work anymore. It was time to step away from that gig and see if there was a way to put some fun back into working with computers.

I entered my "retirement" in a small home office, but found out within a day or two that it wasn't really going to work for me. I needed to put distance between home and my office. Luckily, a potential client offered me a small "rent free" office in the center of town where he already had secretaries, copiers, broadband, and lots of PCs at each desk to tinker with if I was ever in the mood.

This client has his own consultants and contractors to handle hardware, networking, and a lot of his canned software problems, so I can just program away without being bothered with day to day responsibilities. It's that ultimate programmer's dream: give me an office where nobody bothers me, I write some cool stuff, and I deliver in the end without having to put fires out all the time. But even in paradise, there are flaws.

A rent free office means you're a computer expert within easy reach when there is a problem and the business doesn't want to take the time to call the techs they normally call. And a person who managed an enterprise made up of dozens of businesses can't help but try to run the place from time to time. Before I can stop myself, I find that I've jumped right back into the type of work I did before. I'm trying to learn when to back right out, and I'm doing a pretty good job of it, but one misstep suddenly became

this story.

One day, while reading through financials for a client, I began to hear a lot of loud voices in the main office. Being the curious sort, I grabbed my cup and headed down the hall acting like I was only in need of some hot coffee.

The president of the business was in heated discussion with his main computer tech. It seemed that they'd attempted to put new backup procedures into place and, suddenly, they were finding out that the backups neglected to report an error to the end user on a backup failure. The "Ghost of the Vice President of an Insurance Company's Infrastructure" took demonic possession of me and this strange voice came out of my mouth.

"As part of the project I'm working on for you, we will need to compress data, transmit it via the internet, and even write it out to backup media. I'll just write a quick version of it now to solve your backup needs and I'll be able to utilize the same code once we get to that point down the road in the new project."

The president was pleased because this would solve a major problem for him, and he liked my techniques of giving feedback to the end users. The computer tech liked it because it took this horrible responsibility off his back. And all I had to do was see if the bulldog sized check my mouth just wrote could actually be cashed by my Chihuahua rear end.

Compression first

I had experience in the past zipping files up with [Lindersoft's](#) wonderful zip compression tools. This time, I would have to provide a lot of feedback to the end user, so I had to dig a little deeper than normal. The provided classes within the tool did a good job, but I needed just a bit more. Luckily, the classes contain full source and I was able to learn exactly how the library gets things done and, in a little while, that portion was finished.

FTP second

I had experience using [Catalyst's SocketTools Secure Library Edition](#). Once again, I had to dig a little deeper to get lots of feedback on the window for the end user, but soon that was completed. The only thing left was that portion to burn a CD. That's the easy stuff, right? And that's when I hit the wall.

CD Burning

I did some digging around the internet and found out that XP has CD burning built in. That looked like the right answer for this problem as all of the machines involved had burners and ran XP. Also, these backups are maintained for twelve to eighteen months for auditing purposes so single use CD-R was perfect. If this changed down the road, the same XP routines wrote to CD-RW disks. Things were going great and my confidence was soaring. It was time to crow about my sudden brilliance so I dropped a line to my favorite magazine editor and said, "Dave, I'm going to send you some code for the magazine to burn CD-R's. It looks like three or four Windows API calls and I'm finished." So, for those of you keeping count at home, that was the second check my bulldog mouth was writing.

API vs COM

Historically, I've found calling Windows APIs to be pretty easy. I prototype a simple call, add it to my global map, and away I go. Step one in this project was more of the same. There was a single API call that returned the directory to place the backup file so that XP knows that it is the staging area for the CD. Nice and easy as long as you use the correct variable types in your prototypes. But the next part introduced me to a part of Windows that I really had not delved into before.

Microsoft has seen the world make a shift toward more object oriented programming. The original API was procedural and difficult to maintain from Microsoft's standpoint. As Windows has grown up, the complexity of the API has increased and become more object oriented. Multithreading and process communication also has added to the complexity in Windows. So Microsoft worked on specifications for all of this that eventually became the Component Object Model, or COM. And COM is the way to interface to many new API functions within Windows.

Back to the problem

The three functions I needed to finish my program all are contained within the ICDBurn COM interface of Windows. I needed to do some COM stuff to make it work, and I'm no COM expert. It looked complicated and I didn't really understand COM and so, soon, I realized I was in over my head.

I decided maybe the project has gone far enough to make it useable, even though it isn't perfect. I had called the API to copy my backup file to the CD staging area, and Windows XP has a nice built-in wizard that finishes all the burning as long as your end user knows where to click and where to go. The ICDBurn interface would allow me to

pop that wizard up within the context of my code, but I decided I could live with the application not doing that until I learned more about COM. It was time to write up my article utilizing the single API and be on my way.

Beaten to the Punch

Then I got a note from Dave Harms that said, "Would you like to look over this CD burning article?" And suddenly, on my computer, is "[Hunka-Hunka Burnin' CD Data, or Two Steps Forward, One Step Back\(up\)](#)" by a [clucker](#) named Mark Riffey. So now I have a program that doesn't go the last step that I wanted and I have an article in hand that makes mine redundant because Mark had just been through a similar problem. Two checks written, nothing cashed. Well, if you're going to be overdrawn anyway, why not write a third check? I fire off an email to Mark saying congrats on the great article, and that I'll go ahead and take the pain of figuring out the COM stuff. I'm even going to send the stuff to him. Three folks are now waiting on me and the only problem was I didn't have a clue how to do it.

Digging into COM

I read some excellent articles in Clarion Magazine about COM, most of which have been written by [Jim Kane](#). He has a nice set of classes and utilities that help make it easy to prototype your interfaces. My eyes glazed over in places, but as long as I could make it work, it didn't really matter if I didn't understand everything going on under the covers. I dropped a note to SoftVelocity's documentation guys asking if there were any additional documents describing calls to COM within Windows that I was missing. Kindly, they sent me one they are working on that, although it isn't ready for primetime, helped nudge a few brain cells forward.

In order to use a COM object, there first are two things you need. You need the identifier to the COM object itself, and then you need the identifier for the interface you need to use for that COM object. You can think of the COM object as the set of all the functions related to a particular feature within Windows, and an interface as a subset of related functions within the large set.

The next step is to define the interface much as you do a class. The only complicated part here is getting the right data types for each parameter.

After reading through a few pages at [MSDN](#), I discovered that the ICDBurn functions are contained within the shell32.dll, which is part of Windows XP. Jim has a great utility called [RTLlib](#) that helps build all the prototypes for the COM interfaces in a given DLL. I ran it and this great, wonderful file was created. But there wasn't a reference *anywhere*

to ICDBurn. For some reason, Microsoft hasn't "exported" this interface, and there I was stuck again. Where would I find the info?

I found many references to IDL files after searching the web. IDL files are used by C++ folks to talk to COM objects. Microsoft has a free download of the Windows Software Development Kit that includes all of the IDL files needed. I just needed to figure out how to change those variable types into something that Clarion would understand.

It was time for my first attempt.

From the shobjidl.idl file ...

The COM object is defined as ...

```
// CLSID_CDBurn
[ uuid(fbcb8a05-beee-4442-804e-409d6c4515e9) ] coclass CDBurn { interface ICDBurn; }
```

And later in the code, the interface is defined as:

```
[
    object,
    uuid(3d73a659-e5d0-4d42-afc0-5121ba425c8d),      // IID_ICDBurn
    pointer_default(unique)
]
interface ICDBurn : IUnknown
{
    HRESULT GetRecorderDriveLetter([out, size_is(cch)] LPWSTR pszDrive, [in] UINT cch);
    HRESULT Burn([in] HWND hwnd);
    HRESULT HasRecordableDrive([out] BOOL *pfHasRecorder);
};
```

Step one is define the identifiers that Windows uses for the COM object and the interface.

```
CoClassCDBurn      group
Data1              long(0FBEB8A05h)
Data2              short(0BEEEH)
Data3              short(4442h)
Data4              STRING(' <80h><4Eh><40h><9Dh><6Ch><45h><15h><E9h>' )
                  END
IIDICDBurn         group
Data1              LONG(3D73A659h)
Data2              short(0E5D0h)
Data3              short(04D42h)
Data4              STRING(' <0AFh><0C0h><51h><21h><0BAh><42h><5Ch><8Dh>' )
                  END
```

That's the way Microsoft has specified identifiers need to be built, and so, although it is a bit tedious, it is pretty simple to see how you convert the IDL version to the Clarion version. Next is the definition of the interface.

COM objects inherit their behavior from some Windows classes called either IUnknown or IDispatch. The definition shows that this one uses IUnknown so I got:

```
ICDBurn INTERFACE ( IUnknown ) , COM, TYPE
    END
```

One thing contained within the document I got from SoftVelocity was a short but incomplete table on data type conversions from the IDL. As luck would have it, some of the data types I needed I could find from similarly defined examples in Clarion Magazine, but not all of them. From there, Jim was kind enough to correct my other errors with regards to the prototyping. Here's the table with my additions to it.

| COM Type | Clarion Type |
|---------------------|---------------------|
| VARIANT_BOOL | SHORT |
| VARIANT_BOOL* | *SHORT |
| BSTR | BSTRING |
| BSTR* | *BSTRING |
| LONG | LONG |
| LONG* | *LONG |
| UNSIGNED INT (UINT) | UNSIGNED |
| UNSIGNED INT* | *UNSIGNED |
| INT | SIGNED |
| INT* | *SIGNED |
| VARIANT | VARIANT |
| VARIANT* | *VARIANT |
| <i>LPWSTR</i> | <i>LONG</i> |
| <i>HWND</i> | <i>LONG</i> |
| <i>BOOL*</i> | <i>*LONG</i> |

Next, I attempted to define the interface.


```

ICDBurn INTERFACE( IUnknown ), COM, TYPE
GetRecorderDriveLetter  procedure(LONG pszDrive, UNSIGNED cch), HRESULT
Burn                    procedure(LONG hwnd), HRESULT
HasRecordableDrive     procedure(*LONG pfHasRecorder), HRESULT
END

```

This was all great except it wouldn't compile. Clarion has no clue about the IUnknown. And here is where there are two implementations of the solution.

Jim has a nice group of includes that you can add to your application to make it easy to do COM. SoftVelocity also has some includes that do some similar things. I'll show an implementation using some of Jim's stuff first.

The Jim Kane version

To use Jim's classes, you must include his stdcom2.inc, which defines the class that handles standard COM interfaces. Also, there is a small assembly language file called calla.a that his code uses. The last file is the stdcom2.clw file, which is the source for his class. It is automatically included by the stdcom2.inc file.

You'll notice that the interface statement requires INTERFACE(IUnknownType) as this is the name Jim has given to the IUnknown COM interface within his include file. Here's the declaration for the ICDBurn interface:

```

ICDBurnType INTERFACE(IUnknownType), COM, TYPE
GetRecorderDriveLetter  procedure(LONG pszDrive, UNSIGNED cch), HRESULT
Burn                    procedure(LONG hwnd), HRESULT
HasRecordableDrive     procedure(*LONG pfHasRecorder), HRESULT
END

```

In the source code, cdburn.clw, you see the COM class initiation followed by a call to get the interface to ICDBurn.

```

Stdcom2cl.initcom(1)
  lpICDBurn=Stdcom2cl.GetInterface(address(CoClassCDBurn), address(IIDICDBurn))
  if ~lpICDBurn then
    Message('your os does not support burning cds.')
    do procret
  end

```

Then, the code copies the zipped backup file over to the staging area as described in [Mark Riffey's article](#). Now is when I put the ICDBurn interface to work.

The first call checks that the computer has a recordable drive HasRecordableDrive. This is done mostly for illustration of this method as I already know the PC has a recordable drive from Mark's earlier call to a different API.

The next call, `GetRecordDriveLetter()`, gets the drive letter of the recordable drive. This can be useful to prompt the user to insert a CD into that drive. The drive letter is stored as a wide string, but Jim has a function within his class to convert wide strings to Clarion strings.

Lastly, the code calls the `Burn` method. This will pop the XP CD writing wizard up and walk the end user through putting a CD. The wizard will also warn the end user if the burn was not successful.

From there, it was time to clean up and go home.

The SV version

For the SoftVelocity version, I did some different things to better match the examples in the documentation that I was working from. You must include `sncdburn.inc` and `sncdburn.clw` along with the definitions of Softvelocity's COM classes, found in `svcom.inc` and `svcom.clw`. The Softvelocity files are in your Clarion 6 installation in your library source subdirectory, along with all the other classes from Softvelocity.

The `ICDBurnType` interface declaration is the same as the Jim Kane version, except the interface is derived from `IUnknown`, which is how SV declares what Jim calls `IUnknownType`. The helper classes that obtain an instance of this interface are different. First, I defined a class within `sncdburn.inc` that is of the type `CCOMObject`. That allows me to use SoftVelocity's COM methods within my class. In that class, I include three methods that match the `ICDBurn` interface's three methods, along with housekeeping methods of `Construct`, `Destruct`, and `Init`. Lastly, I have a pointer in that class to the COM interface I am going to init. Here's the declaration:

```
CCDBurn          class(CCOMObject), module('sncdburn.clw'), |
    link('sncdburn.clw', _COMLinkMode_), dll(_COMDllMode_)
Construct        procedure()
Destruct         procedure(), virtual

Init             procedure(), proc, HRESULT, virtual

GetRecorderDriveLetter  procedure(LONG pszDrive, UNSIGNED cch), proc, HRESULT
Burn              procedure(LONG hwnd), proc, HRESULT
HasRecordableDrive  procedure(*LONG pfHasRecorder), proc, HRESULT

ICDBurnInClass   &ICDBurnType, protected
                end
```

Within the `sncdburn.clw` file, you see the call in the `Init` class that is similar to the `GetInterface` in Jim's. All the other methods are identical to their COM interface equivalents.

Now to the source code. SV's support classes use a neat trick to initialize the COM library. You simply make sure your code declares an instance of the `CCOMIniter` class (here called `COMIniter`). This class has an automatic constructor that initializes COM, and a destructor that shuts everything down. The example has the following objects declared:

```
COMIniter    CCOMIniter
MyICD       CCDBurn
```

My code first makes sure that `COMIniter` started up successfully, and then initializes the `CCDBurn` instance:.

```
if COMIniter.IsInitialised()
    hr = MyICD.Init()
    if hr = S_OK
```

If that works well, the code checks for the existence of a CD drive with `HasRecordableDrive`, and if that works it gets the drive letter. Following is that bit of code from [Mark's article](#) to copy the file into the staging area. Then the code goes into an accept loop, waiting for a click to initiate the burn process by calling up the wizard. And a single click makes the burn happen.

Note that both of the following equates must be correctly set in the project in order to use SV's COM classes:

```
_COMLinkMode_
_svLinkMode_
```

These equates tell the runtime whether the classes are linked in (which they are in this case) or found in an outside DLL. Without these important switches, you can get a GPF.

In my C6 ABC EXE apps, I have the following set in my project defines:

```
_COMLinkMode_=>1
_COMDLLMode_=>0
_svLinkMode_=>1
_svDLLMode_=>0
_ABCDllMode_=>0
_ABCLinkMode_=>1
```

So now I have an implementation that matches Jim's classes along with an implementation that will match any future examples coming from SoftVelocity.

I have a program that cashes the check I promised with the initial program. I have example code to send to Mark just as I promised. And I have an article for Clarion Magazine just as I promised Dave.

Special Note of Thanks:

The trick to writing checks bigger than you have the ability to pay is knowing who has that ability, and being humble enough to fall on your knees to plead for help.

Thanks to both Jim Kane and Robert Zaunere with SoftVelocity for a lot of handholding and for the examples that made this article possible. One day, when I get big, I want to be a full bird colonel, dentist, architect, and programmer extraordinaire, as well as the president of a software development tools company, and have velvet paintings of me all over the world.

Links

- [ICDBurn reference at MSDN](#)
- [C++ example](#)

[Download the source](#)

Andrew Guidroz II, when he isn't traveling around the countryside watching his 2004 National Champion LSU Fighting Tigers, writes software for all facets of the insurance and finance industries. 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](#)

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Reborn Free

CLARION
online

Generating MS SQL Server Side Triggers

by Ayo Ogundahunsi

Published 2004-04-30

Clarion 6 comes with a lot of new features, amongst which is the capability to integrate client-side triggers as part of the design of the database.

I must say this new feature is quit ingenious, and it is a very bold step. Nevertheless, without downplaying its usefulness, I have always preferred the traditional way of using server-side triggers, because of their inherent advantages. These include:

1. Ability to monitor and audit updates no matter how the database is modified. This means even if a table is updated outside of your Clarion application, you can still audit who made this change. In today's world, scalability and interoperability among diverse components is usually preferred, hence, if you wrap you audit logic within Clarion, you will have to maintain at least a second code base if an external system is going to be updating your database.
2. Centralized management of data integrity, and in some cases referential integrity.

What are Triggers?

The purpose of this article is not to give a detailed exposé on triggers, but rather to present a useful template that can help automate auditing of user updates to your database. Auditing means creating a log of who made what change to a table, and when the change happened. This template

automates the process of creating both the audit table(s), and the triggers which create the audit data.

Since there is extensive documentation in books, or on the internet on Triggers, I will not explain in great detail what triggers are or the kind of triggers available for use.

If you're unfamiliar with triggers, here is a definition from SQL Server's Books Online:

Microsoft® SQL Server™ 2000 triggers are a special class of stored procedures defined to execute automatically when an UPDATE, INSERT, or DELETE statement is issued against a table or view.

Triggers are powerful tools that sites can use to enforce their business rules automatically when data is modified. Triggers can extend the integrity checking logic of SQL Server constraints, defaults, and rules, although constraints and defaults should be used instead whenever they provide all the needed functionality.

Tables can have multiple triggers. The CREATE TRIGGER statement can be defined with the FOR UPDATE, FOR INSERT, or FOR DELETE clauses to target a trigger to a specific class of data modification actions. When FOR UPDATE is specified, the IF UPDATE (column_name) clause can be used to target a trigger to updates affecting a particular column.

For further reading on Triggers, you can check the following resources:

- [MSDN](#)
- [Microsoft Support site](#)
- [DevBuilder](#)

The Template

This template, which was inspired by ideas on auditing with triggers in one

of Ken Henderson's books, will take tables defined in a Clarion Dictionary and generate matching trigger scripts.

The template file is `cmag_SQLTRG.tpl`. You have to register it the same way you register all other templates. Since the template code is contained in a single file, you do not need to update paths in your `.RED` file if you choose to put the template somewhere other than the templates directory.

The template is a Utility template and to use it, you need to have an application loaded. Then choose `Application|Template Utility` from the Clarion main menu, or press `Ctrl-U`. Choose the `sqlServer_AuditTrigger` template, as shown in Figure 1.

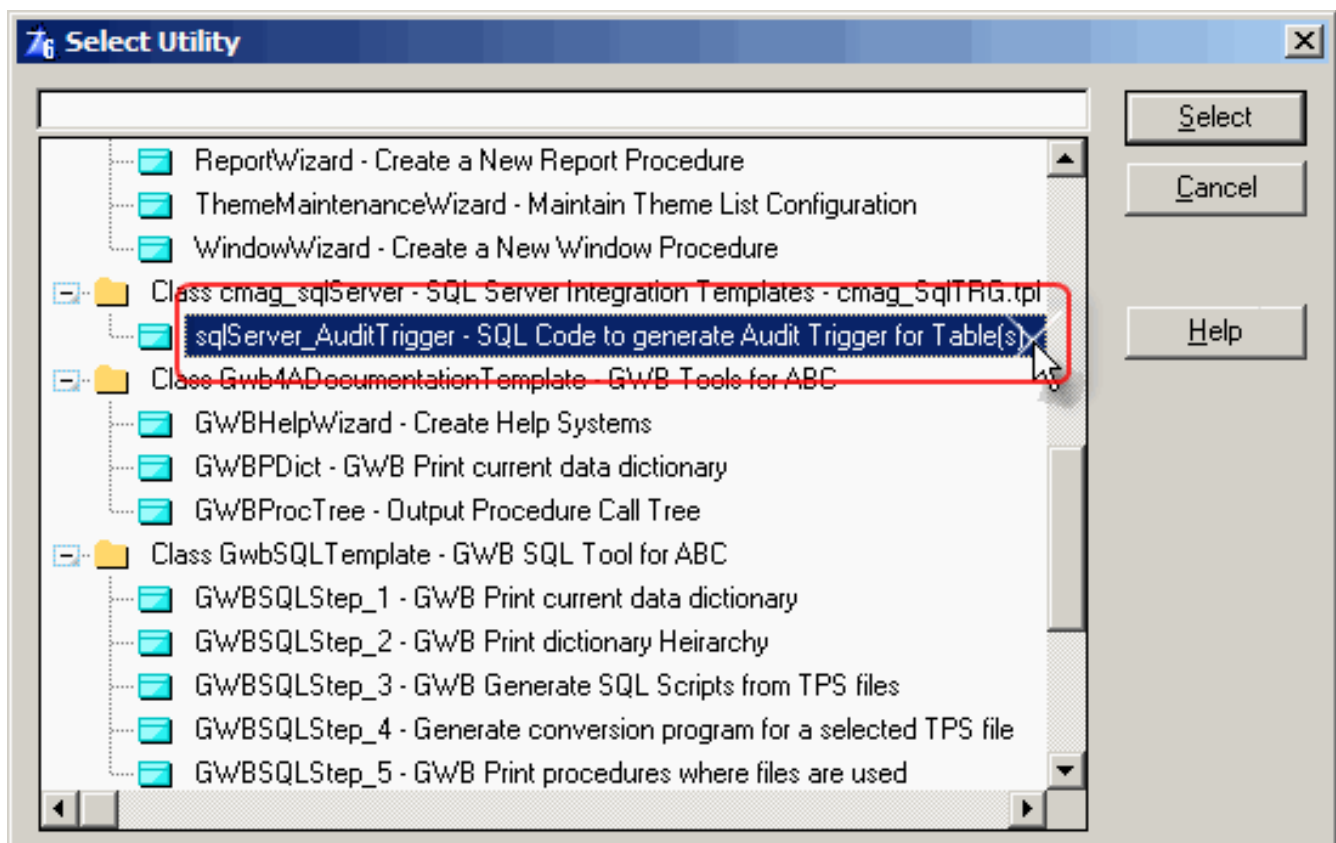


Figure 1. Selecting template

Once you select the template, you will be presented with a wizard, as shown in Figure 2. The tables contained in the dictionary for the application are automatically made available for selection.

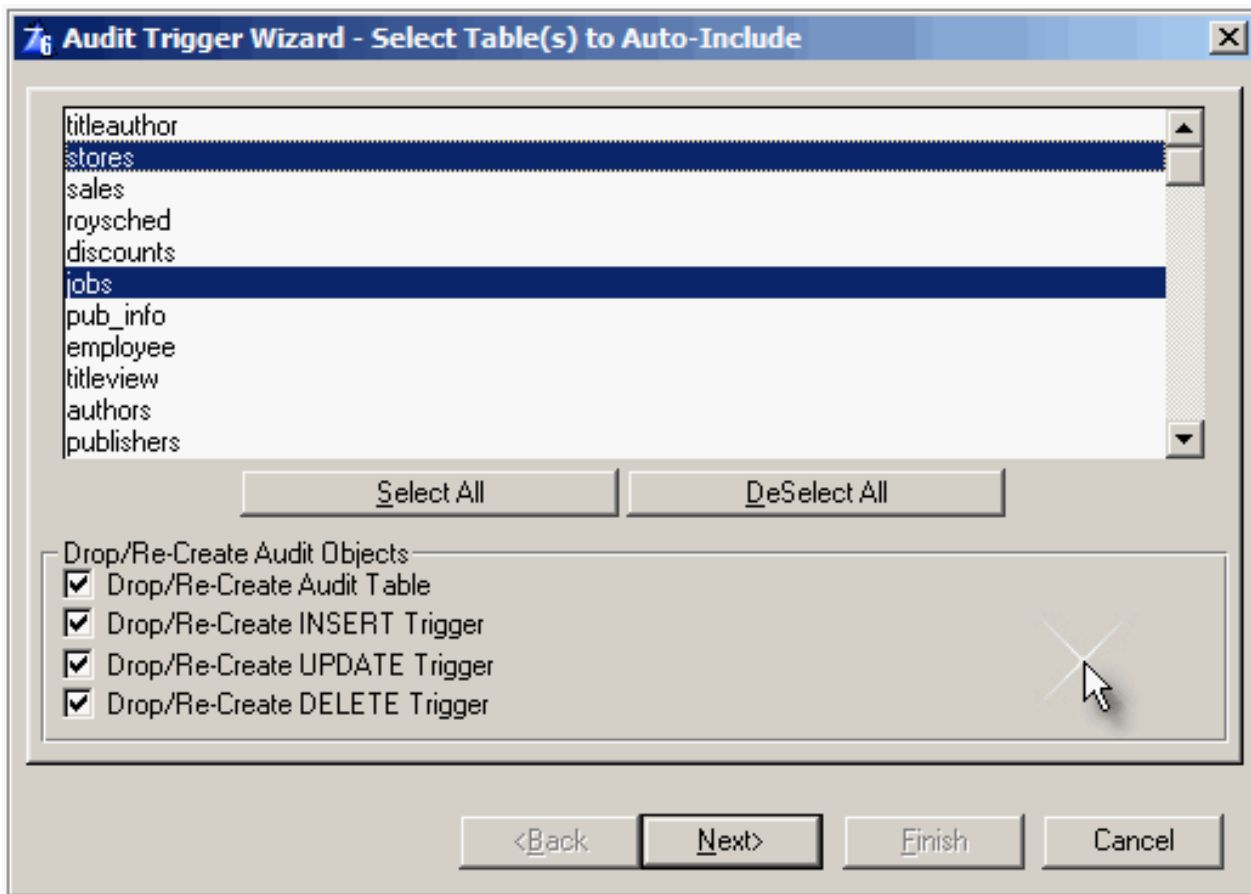


Figure 2. Selecting Tables to Audit

Tables to Audit

This form on the wizard allows you to select the table(s) to create audit triggers for. You can generate all the trigger code into a single file, or individual files that reflect the name of the tables.

Drop/Re-Create Audit Objects

The Drop/Re-create checkboxes simply allows you generate SQL code that will always DROP and CREATE the trigger whenever you run the generated script.

Audit Labels

The Audit label fields allow you to define what entry is recorded in the audit table

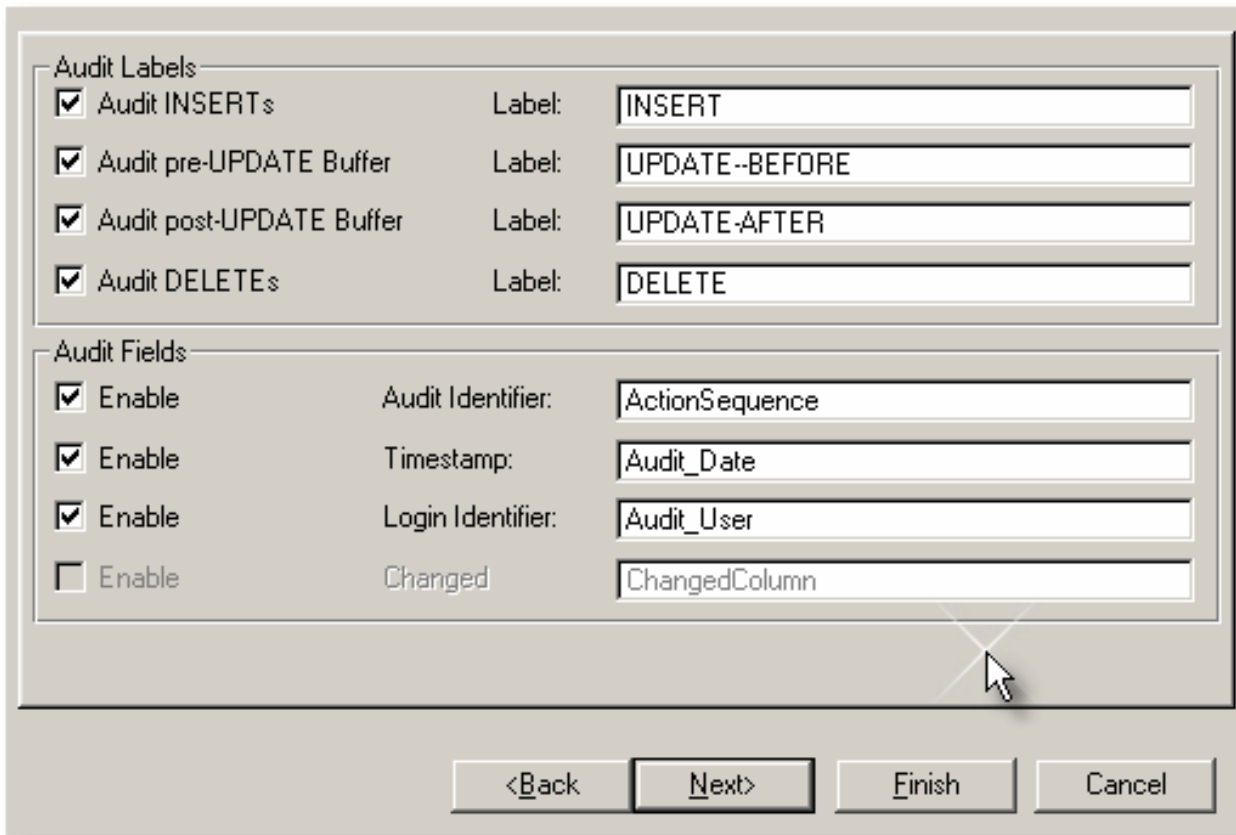


Figure 3. Audit Labels & Fields

As shown in Figure 4, whatever text is typed for the label will be stored accordingly under the ActionSequence column.

Data in Table 'JOBS_AUDIT' in 'pubs' on 'AYO-LAPTOP\AYO_MV01'

```
SELECT *
FROM JOBS_AUDIT
```

| job_id | job_desc | min_lvl | max_lvl | ActionSequence | Audit_Date | Audit_User |
|--------|------------------------|---------|---------|----------------|-------------------|-----------------|
| 3 | Publisher | 150 | 250 | UPDATE--BEFORE | 4/17/2004 6:22:13 | AYO-LAPTOP\Ayod |
| 5 | Publisher - ClarionMag | 150 | 250 | UPDATE-AFTER | 4/17/2004 6:22:13 | AYO-LAPTOP\Ayod |
| 15 | this is a test | 22 | 50 | INSERT | 4/17/2004 9:05:04 | AYO-LAPTOP\Ayod |
| 15 | this is a test | 22 | 50 | DELETE | 4/17/2004 9:06:28 | AYO-LAPTOP\Ayod |
| * | | | | | | |

Figure 4. Representation of Audit Labels

Audit Fields

The concept of the template is based on the fact that you should be able to take a snapshot of a record buffer (or row) before an update, or save a

record buffer (or row) that was used to add a new record (row) to your table.

As a result of this you need to have an audit table with a structure similar to the table you are auditing, however, with three extra fields. These fields are:

1. **Audit Identifier** – Here you store a text that describes the update action you just performed which you are auditing.
2. **Timestamp** – Records the date and time the update action was performed. This is updated with the Transact-SQL `GETDATE()` function.
3. **Login Identifier** – Records the user that performed the update action. This is updated by the Transact-SQL `SUSER_SNAME()` function.

Depending on whether you enable any of these fields (using the Enable checkbox), the script generated may or may not provide code to add them to the audit table.

For example, if you want the `Timestamp` field to be added to your audit table, this is the kind of code the template generates:

```
-- Add 'Audit_Date' column to structure if it doesn't exist already
IF NOT EXISTS
    (SELECT COLUMN_NAME FROM #TEMP_AUDIT WHERE COLUMN_NAME = 'Audit_Date'
     AND TABLE_NAME = 'JOBS')
BEGIN
    ALTER TABLE dbo.JOBS_AUDIT ADD Audit_Date DATETIME
END
GO
```

`#TEMP_AUDIT` table is a table containing a list of fields that belong to an audit table.

You can also change the column names to whatever you desire. For example, you can choose to change `'AuditSequence'` to `'UpdateAction'`.

Script Destination

You can define where the generated script is output to. If you want you can also generate code for multiple tables into a single file, this way it is easier to apply the script once.

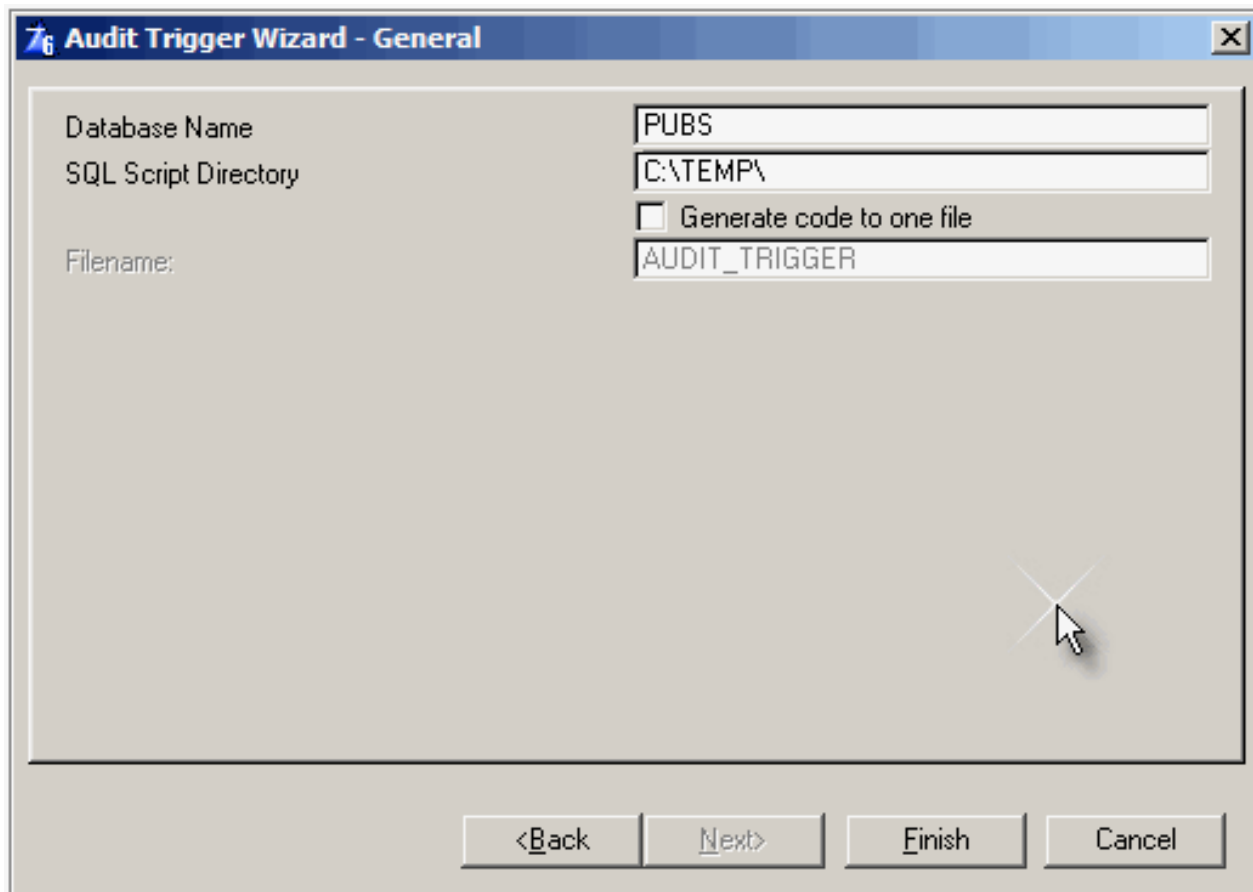
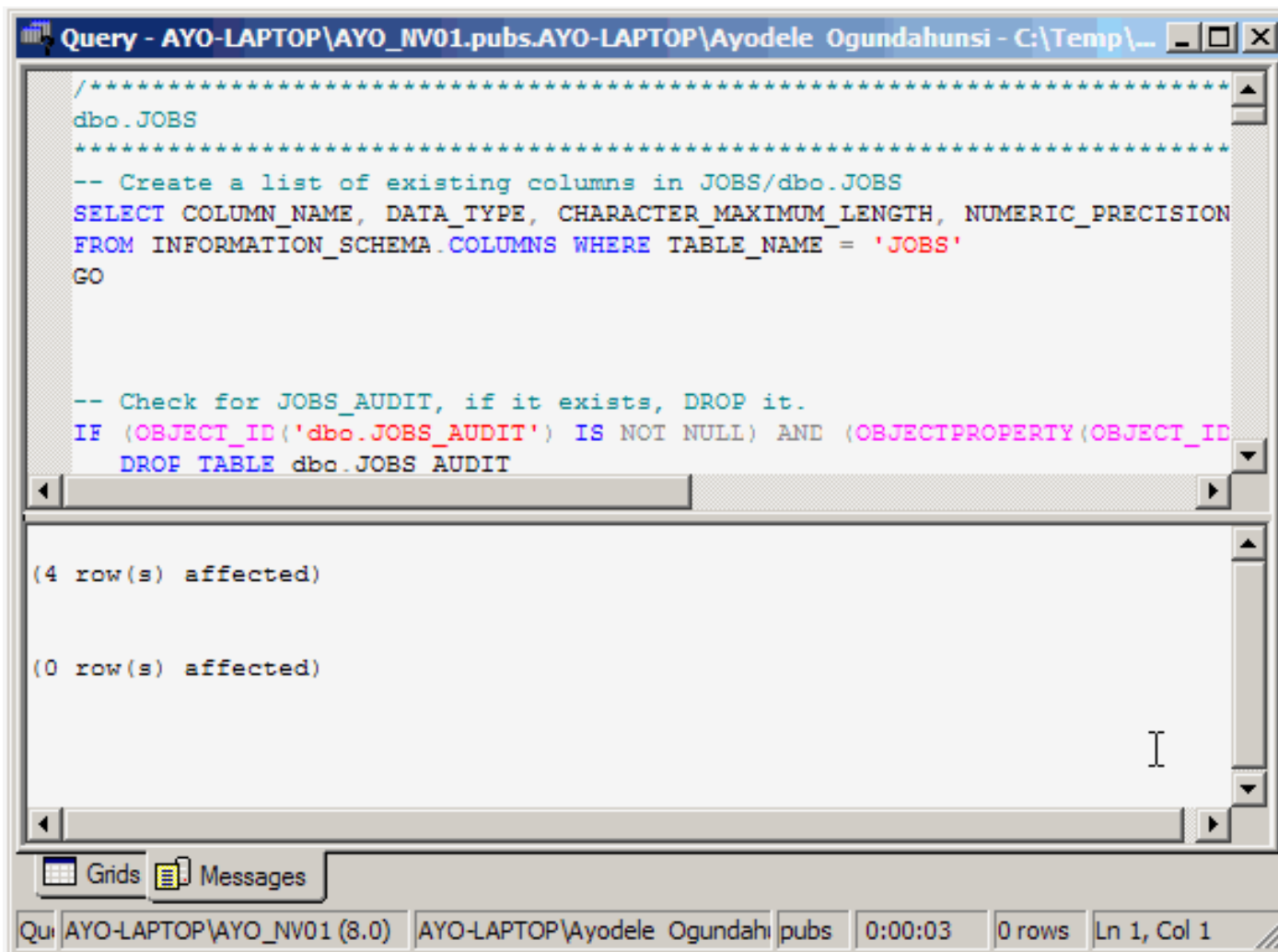


Figure 5. Script Destination

Running the script

It is easy to run the created script. From your Query Analyzer, select the database that contains your tables, and run the script from there.



```
Query - AYO-LAPTOP\AYO_NV01.pubs.AYO-LAPTOP\Ayodele Ogundahunsi - C:\Temp\...
/*****
dbo.JOBS
*****/
-- Create a list of existing columns in JOBS/dbo.JOBS
SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, NUMERIC_PRECISION
FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'JOBS'
GO

-- Check for JOBS_AUDIT, if it exists, DROP it.
IF (OBJECT_ID('dbo.JOBS_AUDIT') IS NOT NULL) AND (OBJECTPROPERTY(OBJECT_ID('dbo.JOBS_AUDIT'), 'IS_TABLE') = 1)
DROP TABLE dbo.JOBS_AUDIT

(4 row(s) affected)

(0 row(s) affected)
```

Grids Messages

Query: AYO-LAPTOP\AYO_NV01 (8.0) | AYO-LAPTOP\Ayodele Ogundahunsi | pubs | 0:00:03 | 0 rows | Ln 1, Col 1

Figure 6. Executing generated script

Once you do this you will see the Audit table created as shown in Figure 7. ActionSequence, Audit_Date, and Audit_User are the three extra fields added to the audit table for JOBS (i.e. JOBS_AUDIT which has a similar structure).

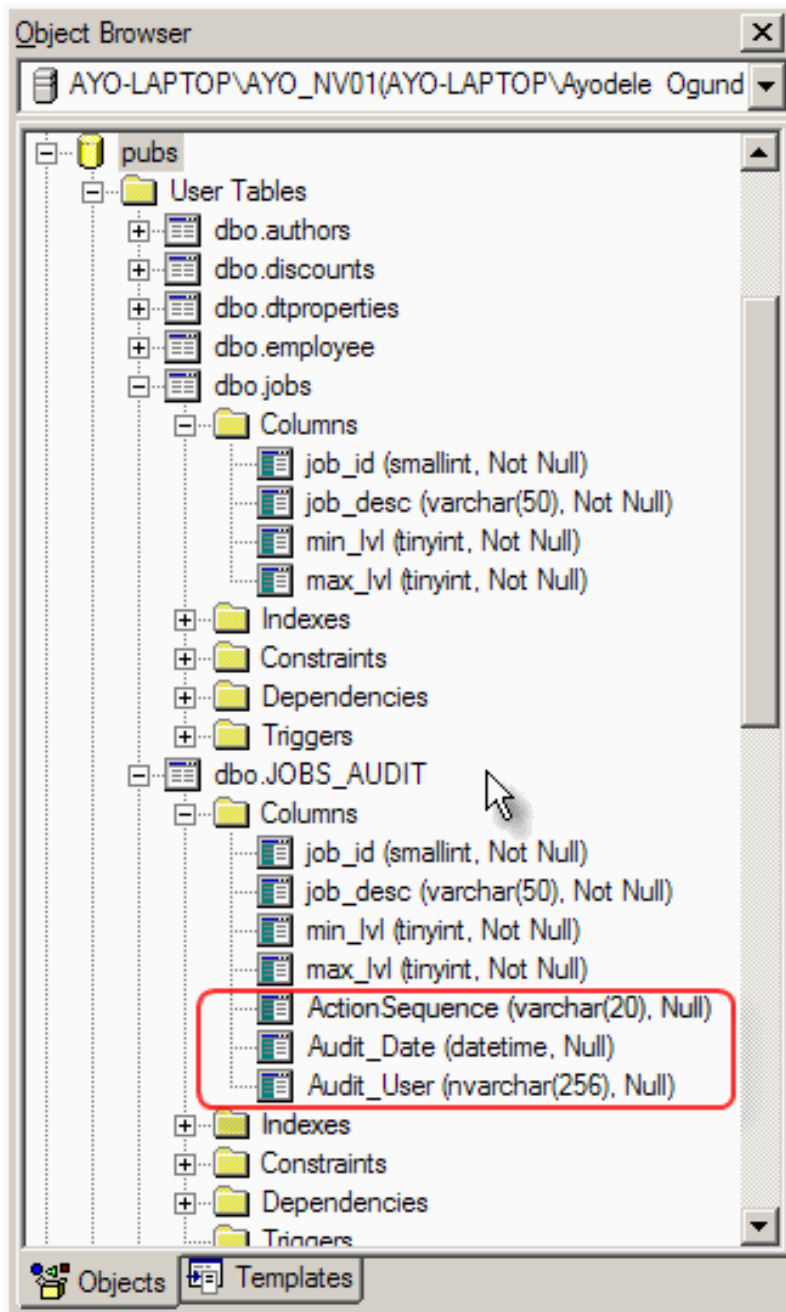


Figure 7. Audit fields in audit table

Figure 8 shows the created triggers. JOBS_INSERT, JOBS_UPDATE, JOBS_DELETE are the three triggers added to the JOBS table.

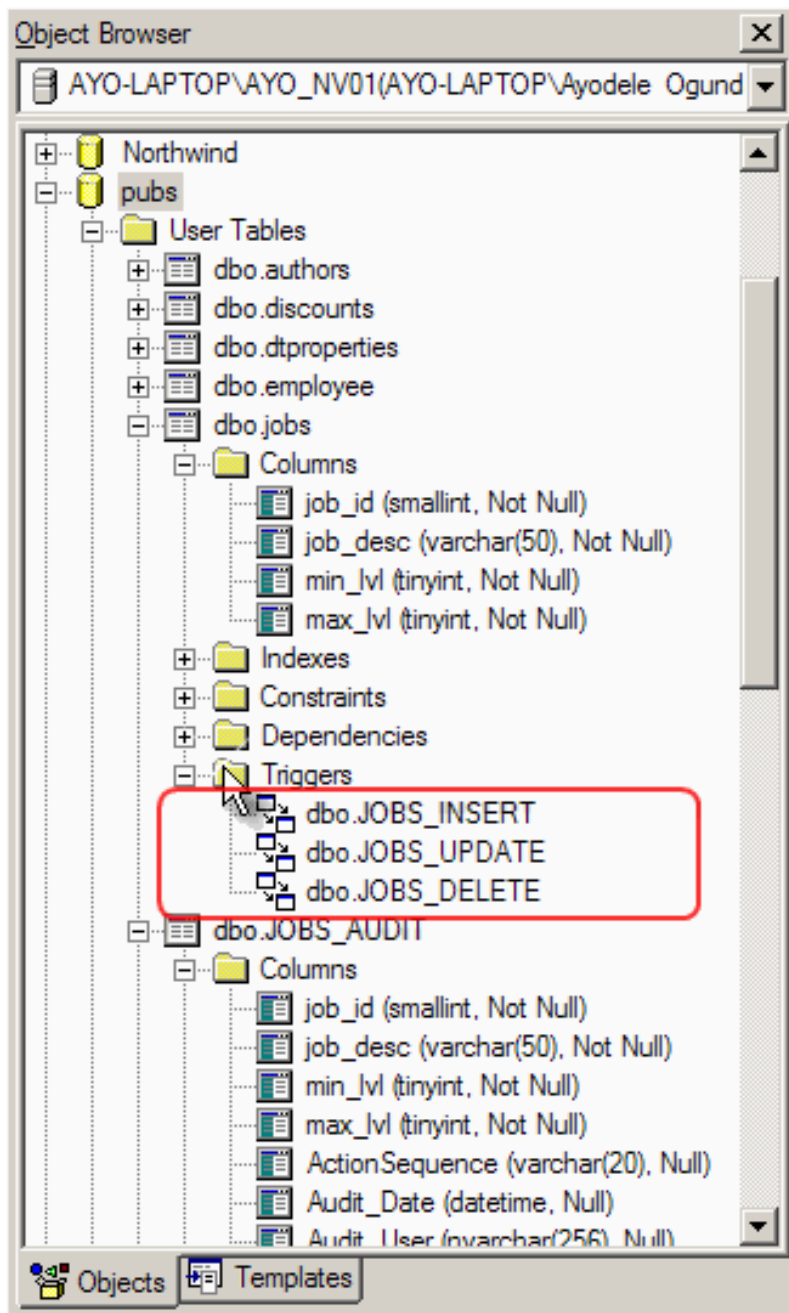


Figure 8. Generated Triggers

How it works

To demonstrate how the applied script works, I've imported the PUBS database, that comes with SQL Server, into Clarion and I've generated an application. I will be using the JOBS table.

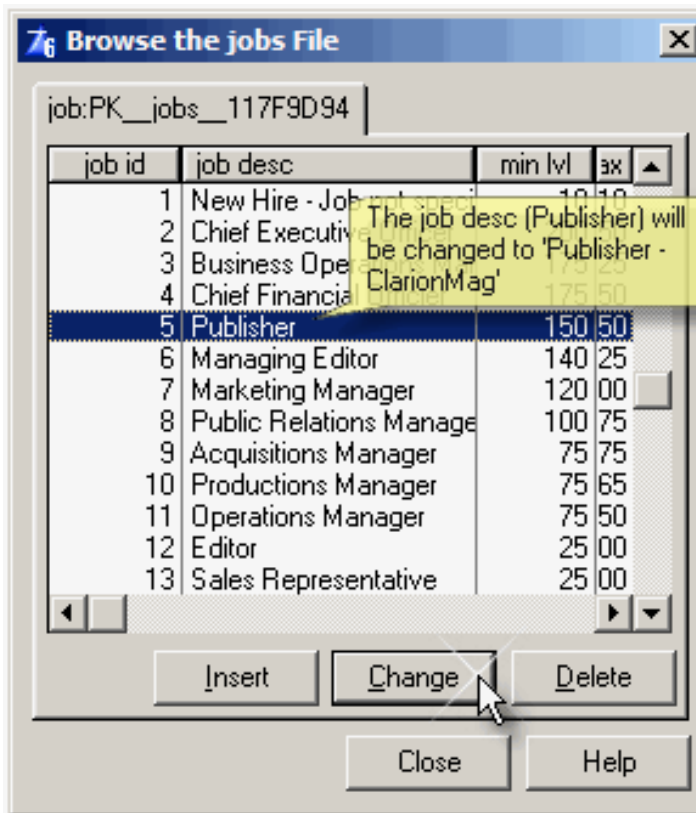


Figure 9. Jobs browse (before update)

Next I load the jobs browse and change the item in the list box with job_id = 5 from 'Publisher' to 'Publisher - ClarionMag'.

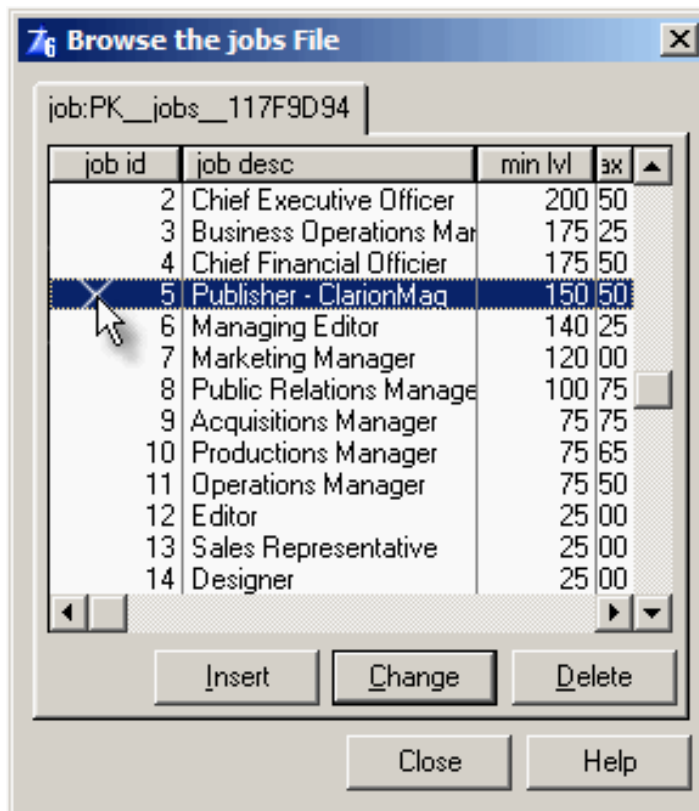
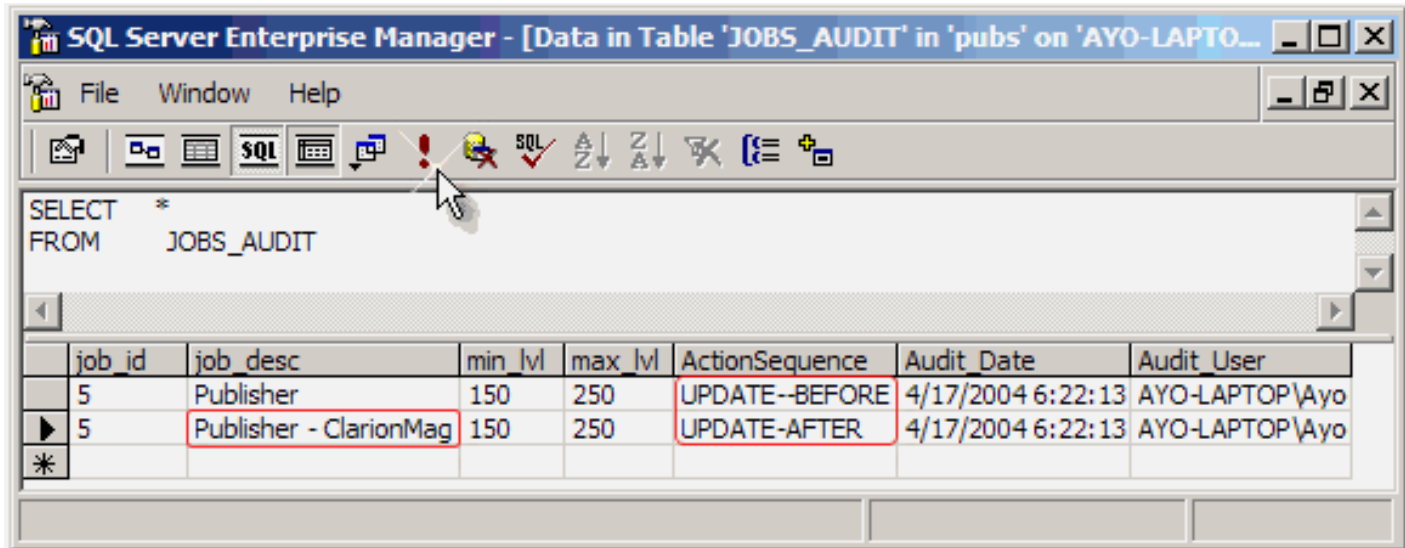


Figure 10. Jobs browse (after update)

I open the table JOBS_AUDIT, and I can see the audit records that have been created as shown in Figure 11.



| job_id | job_desc | min_lvl | max_lvl | ActionSequence | Audit Date | Audit User |
|--------|------------------------|---------|---------|----------------|-------------------|----------------|
| 5 | Publisher | 150 | 250 | UPDATE--BEFORE | 4/17/2004 6:22:13 | AYO-LAPTOP\Ayo |
| 5 | Publisher - ClarionMag | 150 | 250 | UPDATE-AFTER | 4/17/2004 6:22:13 | AYO-LAPTOP\Ayo |

Figure 11. Audit table results (update from Clarion app)

The beauty about maintaining server side triggers is the fact that no matter the application that updates the JOBS table, even if you make a change from the enterprise manager, it logs the action.

In Figure 12, I change the value in the min_lvl column of the JOBS table for the Marketing Manager from 120 to 125 from within the Enterprise Manager;

Data in Table 'jobs' in 'pubs' on 'AYO-LAPTOP\AYO_NV01'

```
SELECT *
FROM jobs
```

| job_id | job_desc | min_lvl | max_lvl |
|--------|------------------------------|---------|---------|
| 1 | New Hire - Job not specified | 10 | 10 |
| 2 | Chief Executive Officer | 200 | 250 |
| 3 | Business Operations Manager | 175 | 225 |
| 4 | Chief Financial Officer | 175 | 250 |
| 5 | Publisher - ClarionMag | 150 | 250 |
| 6 | Managing Editor | 140 | 225 |
| 7 | Marketing Manager | 120 | 200 |
| 8 | Public Relations Manager | 100 | 175 |
| 9 | Acquisitions Manager | 75 | 175 |
| 10 | Productions Manager | 75 | 165 |
| 11 | Operations Manager | 75 | 150 |
| 12 | Editor | 25 | 100 |
| 13 | Sales Representative | 25 | 100 |

Figure 12. Updating jobs table from Enterprise Manager

After doing this, inspection of the JOBS_AUDIT table shows the content of the row before the update (UPDATE--BEFORE), and the new contents of the row (UPDATE-AFTER) as show in Figure 13.

Data in Table 'JOBS_AUDIT' in 'pubs' on 'AYO-LAPTOP\AYO_NV01'

```
SELECT *
FROM JOBS_AUDIT
```

| job_id | job_desc | min_lvl | max_lvl | ActionSequence | Audit Date | Audit User |
|--------|------------------------|---------|---------|----------------|-------------------|----------------|
| 5 | Publisher | 150 | 250 | UPDATE--BEFORE | 4/17/2004 6:22:13 | AYO-LAPTOP\Ayc |
| 5 | Publisher - ClarionMag | 150 | 250 | UPDATE-AFTER | 4/17/2004 6:22:13 | AYO-LAPTOP\Ayc |
| 7 | Marketing Manager | 120 | 200 | UPDATE--BEFORE | 4/19/2004 4:38:16 | AYO-LAPTOP\Ayc |
| 7 | Marketing Manager | 125 | 200 | UPDATE-AFTER | 4/19/2004 4:38:16 | AYO-LAPTOP\Ayc |

Figure 13. Audit table results (update from Enterprise Manager)

Summary

Although this template makes it easy to use triggers for an audit trail, you

need to be aware of potential performance issues. Triggers are actually stored procedures that execute whenever an INSERT, UPDATE, or DELETE happens. If you are doing batch operations on a file, the trigger will execute for each row that is changed, and that could slow down processing. For example, if you have a table you import a large amount of data into, this table should not have the audit trigger created for it.

Having said this, it is up to you to test and determine the effect of using the audit trigger code generated by this template in your peculiar environment.

[Ayo Ogundahunsi](#) presently works for [Impac Medical Systems Inc.](#), the leading company in cancer therapy software. Ayo is married to Ayodola, and they have two boys, Darren and Joshua.

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.