# Clarion Magazine

Articles: XML
News: XML

## PDF for August-September, 2004

All Clarion Magazine articles for August-September, 2004 in PDF format.

*Posted Friday, October 01, 2004*

## DevCon 2004: Implementing CVS Version Control and Clarion 6.x

Rick Martin gave an enlightening DevCon presentation on using the CVS version control system (VCS) with Clarion. Dave Harms reports.

*Posted Tuesday, October 05, 2004*

## DevCon 2004: Business Rules, Triggers and the Local Class

By a quick show a hands, Nik Johnson verified his suspicion that very few people are using the new Business Rules features in Clarion. He reminded us that, as database programmers, we strive for "Third Normal Form" for our data structure designs. Considering the benefits of that philosophy, he challenged us to recognize that the same benefits should be applied to our validation code as well.

*Posted Tuesday, October 05, 2004*

## DevCon 2004: Rebasing and Binding

As Carl Barnes pointed out in his

## News

BST 3.65

dpQuery 2.00

cpTracker Pro Sale

CoveComm Inc. Named Third-party Vendor of the Week

NetTalk v3.20 Gold

SafeWriter v2.12

EasyCOM2INC 1.15

Week 5 Winner of the Clarion Developers Challenge

SetupBuilder 4 - SetupBuilder 5 Information

Annual German Clarion Developer Meeting

Ingres Open Source Database

Clarionfoundry Change

Expert Rules Beta Site

ABC Free Templates and Tools Updated Oct 4, 2004

Free Icons Promotion

GWBSQLAutoInc Updated

BoxSoft Super Stuff 6.5

**SURVEY**

**What kind of coffee do you usually drink?**

○ Vending machine

○ Instant

○ Packaged ground beans

○ Home ground beans

○ Store ground beans

○ Coffee shop

○ I don't usually drink coffee

Vote

Previous Surveys

Wednesday DevCon presentation, the full benefit of the EXE/DLL structure can be realized only if the developer understands the process of loading, linking and using these modules at run time.

*Posted Tuesday, October 05, 2004*

## DevCon 2004: Report Generation - Tricks & Tips

Diego Borojovich's session was the last of the conference, and had to be squeezed into a sub-30 minute slot. Still, Diego covered a number of tips for designing reports that can appear similarly in a variety of file formats, including PDF, XML, and HTML.

*Posted Tuesday, October 05, 2004*

## DevCon 2004: Do It Yourself ABC Classes and Templates

This session by Mike Hanson covered the steps required to make your own class ABC compatible and how to write a template so your class shows up in the Embed tree just like the regular ABC classes.

*Posted Tuesday, October 05, 2004*

## DevCon2004: Metadata Management for Increased Interoperability in an Enterprise Database Environment

Mike Gorman presented his database design and development methodology at DevCon 2004, and made a powerful case for a rigorous methodology to promote successful outcomes. His message, presented with good humor

Released!

JAGUAR 1.0 Beta 2

CWPlus ver 3.01

ThinkData's 30% Off Sale Continues Through October 15

Nice Touch Version 6 Upgrades

EasyCOM2INC 1.14

C6.1 HotFix 9028

IMDD HotFix 9028

Clarion Developers Challenge Week 3 Winner

cpTracker Pro Sale

NET Tools Library In PDF-Tools

Blowfish Encryption Code

LANSRAD Software Named Vendor of the Week

EasyResizeAndSplit 2.06

**Search the news archive**

## One Year Ago In CM

Book Review: .NET Framework Essentials

Data Structures and Algorithms Part XXIV - Floyd's All Pairs Algorithm

Clarion 6 First Look: The Source Code

## Two Years Ago In CM

and supported by examples and detailed visuals, was well received by a large audience of DevCon attendees.

*Posted Tuesday, October 05, 2004*

## DevCon 2004: Threading - Just What You Need To Know

If you are still using Clarion 5.5 to develop new applications because you are scared by the new threading model then Steve Parker's DevCon presentation has the solution for you.

*Posted Thursday, October 21, 2004*

## Implementing a Critical Section: Fast and Effective

The Critical Section is the construct most commonly used to protect reading and writing non-threaded, static data in multi-threaded applications. And using Critical Sections is easier than you think.

*Posted Thursday, October 21, 2004*

## Using Clarion's Business Rules, Part 1

Nardus Swanevelder begins a new series of articles explaining how to use Clarion's business rules capability to make your data validation flexible and configurable.

*Posted Thursday, October 21, 2004*

## All Our DevCon 2004 Coverage In One Place

Looking for links to all our DevCon 2004 articles? Check out this page in the topical index.

*Posted Thursday, October 21, 2004*

PDF for October, 2002

Converting The Inventory Example - Calling Stored Procedures (Part Two)

Converting The Inventory Example - Calling Stored Procedures (Part One)

## Three Years Ago In CM

Optimizing DLL Loading - Introduction to Rebasing

Controlling Printers With DevMode (Part 2)

The Clarion Magazine Free Sampler

## Four Years Ago In CM

Give It a Nudge: Adjusting Report Position at Runtime

Clarion News - October 2000

The Clarion Advisor: Copying Browse Boxes Between Procedures

## Five Years Ago In CM

The Other Way To Use OLE - Part 3

Clarion 5.5 Preview

Stephen Mull's Guide To Converting To MS-SQL

## ClarionMag Subscription Price Increase Dec 1, 2004

Changes in foreign exchange rates over the last two years are making it necessary for Clarion Magazine to raise its subscription rates, effective December 1. If you have an existing subscription, you may want to consider renewing before the price goes up!

*Posted Monday, October 25, 2004*

## Using Clarion's Business Rules, Part 2

In Part 2 of this series, Nardus Swanevelder explains the inner workings of the RulesManager class.

*Posted Friday, October 29, 2004*

## Critical Procedures: Synchronization for the Lazy

To protect data that's shared across threads, you can use a Critical Section, but if you fail to release the Critical Secition when you're done with it, you have a Critical Problem. The solution: use a Critical Procedure instead.

*Posted Friday, October 29, 2004*

## Extending the FileManager Class: Shorthand Fetches

One of the most commonly used commands in Clarion is GET, which can retrieve records from both files and queues. The ABC equivalent is the FileManager.Fetch method, that isn't necessarily easier to use. Mike Hanson shows how to make your fetches quick and easy.

*Posted Friday, October 29, 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.

Clarion Magazine

# DevCon 2004: Implementing CVS Version Control and Clarion 6.x

**by David Harms**

Published 2004-10-05

Rick Martin (not *Ricky* Martin, in case anyone was wondering – Rick got applause for promising not to sing, dance, or take off his shirt) gave an enlightening DevCon presentation on using the CVS version control system (VCS) with Clarion. I have to admit that I'm one of those developers not currently using a version control system, although I have done so in the past.

Rick began with a brief explanation of CVS, which is short for Concurrent Versions System. CVS is a commonly-used open source (and multi-platform) version control system. Basically, version control is source file change tracking, which allows you to go back to previous versions of source files, as well as compare versions (CVS gives you the option of using the compare tool of your choice, and Rick's choice is from Araxis). Rick argued that over a long period of time, tracking changes strictly via comments causes the code to become too cluttered. Version control allows you to make changes without fear that you're obliterating your old code. It also allows you to create branches of your code, so you can easily build different versions of your apps for different customers, a feature Rick emphasized as being particularly useful.

Although CVS is often used in multi-developer situations, individual developers can also benefit greatly. CVS handles concurrency, and if two developers check in the same code, the second developer is notified of the potential conflict, and that code is merged into the code already checked in by the first developer.

CVS is command line driven, for compatibility with various make utilities. The WinCVS system is an open source GUI front end to CVS, available for Windows and other platforms. As of WinCVS 1.3, CVS is included with the install. CVS can be server-based, running on a wide variety of platforms. Two advantages are possibly better performance, and remote access. Otherwise there are no differences, and Rick's demonstrations all used a locally mounted CVS.

Rick went on to define some terms. The repository is where the versions of your software are kept. The working files and folders are local on your machine, and you tell CVS that these are under its control. But you don't check in and out app files – instead, you check in/out individual modules (as TXAs) using the Clarion version control interface. CVS really doesn't know anything about your APP files per se. The same applies to the dictionary – you need to export the TXD.

Rick strongly recommended marking all files in CVS as read-only, meaning you have to check those files out to make any changes. This is not the same thing as a lock, but it means you can always tell who is editing which file.

The first step Rick takes is to check in all the shipping Clarion templates plus any third party utilities and import those. He does sometimes modify the shipping templates, and one benefit of this approach is that he can now migrate those changes up to new releases in a fraction of the time it once took. Even if you don't modify the Clarion templates or source code, you may need all this information to correctly recreate the build environment if you one day want to go back to a previous release.

You need to do some work to get Clarion to work correctly with CVS. In Clarion 6 the IDE interface to version control systems is configurable, but unfortunately most of the C6 default settings are wrong. For instance, the correct check in command is "commit", not "checkin". And within Clarion's VCS interface, when using CVS you use Checkin, Add, and Delete, but you never use the Checkout option. Since modules are the smallest part of an app Clarion checks in, Rick puts one procedure in each module, and renames each module to *procedurename_appname*. The biggest thing to watch out for on a check in is that you not overwrite your entire app.

The check out capability (refresh an app's module from the repository) is somewhat dangerous in C6. If you check out a single module, your entire app is refreshed, and you will lose work that has not been checked in. (Actually the problem is that there is no DDE command to import a single module, such as there is in the IDE.) When you do check out the app, global embeds are duplicated; the incoming embeds are orphaned, and the old ones remain, so you don't get changes from the repository. Also global extensions with the MULTI attribute are duplicated. Rick demonstrated a utility (despite a small gremlin - it worked last night!) that is a replacement for the application check out capability, and which will be available to conference attendees, along with Rick's updated presentation notes.

**Audio:** Rick Martin describes how you can build an entire independent make system that integrates with version control.
Windows Media, 334K

When asked if SV was working on the CVS bugs, Rick Martin said he didn't know, but he had reported the problems a number of times. As the session broke up Robert Ricketts said SV would be addressing the CVS shortcomings, although I doubt many people heard him. If you have an interest in using CVS with Clarion, I suggest you let SoftVelocity know, so the fixes remain a priority.

Rick's presentation was packed with information, and there were definitely some points I missed the first time around. SoftVelocity has announced that a DVD of the conference proceedings will be available, and this presentation will be one of the first I will want to review.

---

*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

Clarion Magazine

# DevCon 2004: Business Rules, Triggers and the Local Class

**by Mike Hanson**

Published 2004-10-05

By a quick show a hands, Nik Johnson verified his suspicion that very few people are using the new Business Rules features in Clarion. He reminded us that, as database programmers, we strive for "Third Normal Form" for our data structure designs. Considering the benefits of that philosophy, he challenged us to recognize that the same benefits should be applied to our validation code as well.

## Dictionary

Clarion has a number of methods for handling data validation. The first is Dictionary-based code. You can specify options like "Cannot be zero or blank", "Must be in list", etc. However, Nik urged that these were not enough.

You can also place triggers into the dictionary, but these are used only in ABC (or if you decide to write some of your templates for Clarion legacy). Additionally, you type the trigger code into a regular entry field, rather than the Embeditor, and then you must verify that the code appears in a logical place in your generated source.

The third option is to use the Validator template. It enables you to suppress the standard generated code, either absolutely or selectively; you can add generated code to enhance user friendliness; and you can generate code for "VALIDATOR" user options in the dictionary.

A key problem with all of the dictionary options is that the user does not always get clear feedback from the system.

## File Manager

You can also place your authentication code directly into the File Manager. However, the references are to obscure "field numbers", code generated into embed points must work with other code generated into the same embed, and you don’t have access to the Embeditor. Even so, there are situations where this can come in handy.

## Rules Manager

Nik explained that the new Rules Manager:

- Centralizes business rule definition
- Applies to fields populated anywhere in application
- Enables/Disables fields when rules are broken
- Serves as basis for power and versatile business rules system

One of the key benefits involves friendly feedback for users, including icons beside bad fields, lists of rules that are currently broken, jumping to the invalid field, etc. It’s implemented as a global template in your APP.

When you specify a rule for a field, that rule will be enforced wherever that field appears in your applications. You can specify any expression, and it will be evaluated at run-time. User-defined functions are also supported, providing that they satisfy the limitations of BINDing.

If this is a multi-DLL system, then you put it into your base APP, and it "broadcasts" the rules to the rest of the APPs in your system.

Nik continued by introducing the concept of a standard procedure to contain all validation code. It’s callable by both the Rules Manager and File Manager, supports multiple tests for a single field, and multiple levels of severity. It’s also Embeditor friendly.

There’s an additional system enabling users to add their own rules (or for the developer to tweak the business rules after-the-fact).

He then reiterated that the Business Rules provide a "Normalized" approach to validation code, emphasizing the strengths of uniqueness (all validations use the same code), independence (validation is distinct from the input forms and batch processes), and consistence (the same validation can be triggered from the windows or File Manager, with the same results).

He finished by taking us through the conversion of a typical file-specific, hand-coded validation ROUTINE into a generic user-defined global CLASS. Overall, Nik’s presentation was a good primer for the benefits and implementation Business Rules, and as a result, I expect that more developers will be making use of this powerful feature in the future.

---

*Mike Hanson is President of BoxSoft Development Inc, which produces the "Super" series of templates, distributed through Mitten Software. He has been creating add-on products for Clarion since his Public Domain Models for CPD 2.0 back in 1988. He's also written articles for every Clarion-related publication, and has spoken at numerous conferences and training seminars. If you have any questions, you can reach him via www.boxsoft.net.*

## Reader Comments

Add a comment

Clarion Magazine

# DevCon 2004: Rebasing and Binding

**by Nik Johnson**

Published 2004-10-05

For all but the smallest applications, a common practice among Clarion (and other) developers is to arrange the component procedures and functions into a number of modules, a single EXE and a number of dynamic load libraries, DLLs. In Clarion this allows the use of multiple APP files of manageable size rather than a single APP, with benefits ranging from shorter procedure trees to faster generation and compilation.

Where multiple Clarion EXEs run on the same computer, this also allows sharing of the DLLs which make up the Clarion run-time library saving disc space and (potentially) memory.

However, as Carl Barnes pointed out in his Wednesday DevCon presentation, the full benefit of the EXE/DLL structure can be realized only if the developer understands the process of loading, linking and using these modules at run time.

Win32 binaries (EXE, DLL, OCX) are constructed so that they can be read from disk and used with little modification. These files are known as Portable Executable (PE) files. A PE file on disk is almost identical to the image in RAM. Because of this, the Virtual Memory Manager can discard pages when the memory they occupy is needed for another purpose and reloaded from the disk file, provided that the file was not modified when it was loaded into memory.

And there's the rub. All PE files are linked to load at a preferred address in the virtual memory space. If two or more PE files, loaded at their preferred address, would use the same area, all but one must be modified to load at another address. This modification process is called a "rebase."

Once a file has been rebased, it loses some valuable characteristics:

1. Pages can no longer be reloaded from the original file, so they must be written to the page file, increasing overhead and consuming virtual memory.
2. Multiple processes which use the same DLL can no longer share a single copy of that DLL.
3. The rebasing process increases the time necessary to load the process.

Carl proceeded to describe and demonstrate how the developer can control the preferred addresses of linked modules, including, where appropriate, third-party DLLs. By properly specifying preferred addresses, significant benefits can be achieved, particularly where multiple copies of a process are run, as is the case under Citrix, Terminal Server, and ClarioNet, where server memory usage by each user can substantially reduced.

When a PE file references functions in another PE file, the addresses of these functions must be looked up during the loading process. If rebasing has been done properly, the looked-up addresses will always be the same. Therefore it is possible to speed the load process by looking up and storing (binding) these addresses in advance. Carl described the tools and processes by which this can be done.

In conclusion, Carl reviewed the experience of several Clarion developers with rebasing and binding, including his own. All had achieved substantial gains in efficiency with no adverse side effects.

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

## Reader Comments

Add a comment

**Thanks for the excellent write up Nik. I covered all...**

Reborn Free CLARION online

## Clarion News

Search the news archive

### BST 3.65

Big Schedule Tamer Suite version 3.65 is now available from Comsoft7. This release added embed points for drop, insert, change, delete schedulers, changed routines for 'change' and 'delete' to use SysId in queue, added embed points for drop, insert, change, delete in Wall calendars, changed "use BST PropSql" default to "off" and added embed points for drop, insert, change, delete bookings.

*Posted Monday, October 25, 2004*

### dpQuery 2.00

dpQuery 2.00 is now available. This is a tool which includes libraries and a template that adds a powerful interactive resource (Query wizard) for importing of practically any external data into your program. Having spent only a couple of minutes on adding dpQuery to the application, the programmer provides the end users with an easy to use tool with an intuitive interface, which allows loading data from any source, whether it be xbase files, MS Excel books, MS SQL tables, CSV, Firebird/InterBase, DSN source, MS Access, etc. The user is given a possibility to build simple and complex (relational) queries, to filter and sort the data, as well as edit and run an edited query right away. It allows any user familiar with the SQL SELECT statement to build his own queries of any complexity, which can be saved and invoked at any time.

*Posted Monday, October 25, 2004*

## cpTracker Pro Sale

cpTracker Pro is now on sale for Clarion developers through the end of October, at $69.00 per user. This is $30 off the regular price of $99 per user. Go to www.berthume.com/purchase.htm and select the Custom Solution order option and enter the sales price. cpTracker Professional will assist you in managing your contacts, prospects, customers, sales, projects and tasks to the last detail.

*Posted Monday, October 25, 2004*

## CoveComm Inc. Named Third-party Vendor of the Week

Dave Harms and CoveComm Inc., publisher of Clarion Magazine, have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week! Dave and Clarion Magazine are awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of the Clarion Magazine book "Clarion Databases and SQL," or a one-year subscription to Clarion Magazine! Both items are a great asset to any Clarion Developer's toolchest. If you don't already on these fine publications, get them both! Clarion Databases & SQL is a softcover book containing selected articles on the subjects of databases and SQL from Clarion Magazine's first five years in publication. These articles were written for Clarion 5.0 and 5.5, but many are just as applicable to Clarion 6. Topics include database design concepts, TPS file techniques, introductory SQL, specific techniques for databases including MySQL, PostgreSQL, MS SQL, and Oracle (many of the concepts discussed are transferrable to other SQL databases), SQL book reviews, ABC core class design notes (by David Bayliss, the original author of these classes), and a collection of database tips and techniques.Softcover, approx. 600 pages. Edited by David Harms. CoveComm Inc. ISBN: 0-9689553-3-9. Clarion Magazine is an electronic magazine, published at www.clarionmag.com which now includes well over a million words of Clarion-related information. One of the cardinal rules of software development is don't build what you can buy. You wouldn't think twice about spending the cost of a subscription on a third-party product that saved you hours of work and helped you provide a better application to your customers. And that's exactly what Clarion Magazine is: a product that saves you time and improves the quality of your work. You get how-to articles, source code, tips and techniques, product reviews, Clarion news, and more. You can even exchange comments with authors and other readers via the reader comment link at the bottom of most article pages. As with any other good quality third-party product, Clarion Magazine offers a terrific return on investment! For further information about Clarion Databases and SQL and other news from Clarion Magazine please go to: http://www.clarionmag.com/index.html

*Posted Monday, October 25, 2004*

## NetTalk v3.20 Gold

NetTalk v3.20 has been designated a Gold release. (Development will still continue, but NetTalk has reached a milestone and this is a great place for everyone to upgrade to this latest and greatest release.) New features include: SSL support for HTTPS (Secure HTTP), Secure Email, and SLL NetSimple Client; Asynchronous open as the default behavior for all NetSimple and NetTalk objects; A new web server object that allows your application to have a Web Interface, thereby allowing your application to be accessible on the Internet or over a LAN; Windows 2000 fix for NetSimple objects; Important improvements to the Email and Web objects; Dial-Up Networking fix for Windows XP; Improved Global Template, that now includes more Log File options; As usual the netdemo.app (example) has also been updated to include the relevant changes; Plus numerous improvements, tweaks, optimizations etc.

*Posted Tuesday, October 19, 2004*

## SafeWriter v2.12

Safe Writer is used to encrypt files (using the 168-bit Triple DES Algorithm) and in so doing to create a Safe File, which can then be safely distributed with virtually no risk of unwanted users reading the files. A password or key provided by the user is used for the encryption.

*Posted Tuesday, October 19, 2004*

## EasyCOM2INC 1.15

EasyCOM2INC version 1.15 is now available. Changes include:

Generator supports of the dispinterfaces; Generator supports of IID for coclasses; You can now type the Clarion data types in "Edit data types". Free upgrade for all registered customers.

*Posted Tuesday, October 19, 2004*

## Week 5 Winner of the Clarion Developers Challenge

First place for Week 5 of the Clarion Developers Challenge goes to Dave Sperling! Dave finished in a four-way tie with Cesar Carmona, Jim Henry, and Gregory Bailey, all with 10 correct picks. Dave pulled out the win in the Monday night tiebreaker by coming closest with a total score of 57 (actual total score was 75). For finishing first in this week's Clarion Developers Challenge, Dave will be receiving a copy of a copy of the BMT Big Menu Tree templates, donated by Bo Schmitz and Comsoft7.

*Posted Tuesday, October 19, 2004*

## SetupBuilder 4 - SetupBuilder 5 Information

Lindersoft will stop selling the SetupBuilder 4 Installation System on October 31, 2004. The special SetupBuilder 5 Installation System offer ends on October 31, 2004. All SB5 prices can be found in the new price list for October 2004. Buy SetupBuilder 4 Standard Edition ($199.95) or Web Edition ($299.95) now and get SetupBuilder 5 Professional ($249 value) or Developer ($399 value) free. You'll get SetupBuilder 4 and access to the SetupBuilder 5 (beta) program.

*Posted Tuesday, October 19, 2004*

## Annual German Clarion Developer Meeting

The annual meeting of German Clarion developers will take place on November 19 and 20 in Munich at the Holiday Inn, Unterhaching. The cost will be approx. 60 Euro. Hotel and travel has to be paid individually, if required.

*Posted Tuesday, October 19, 2004*

## Ingres Open Source Database

Computer Associates has released its Ingres SQL database as open source. Ingres is one of the oldest relational databases, with a history spanning some 30 years.

*Posted Tuesday, October 19, 2004*

## Clarionfoundry Change

There have been several substantial changes in Clarionfoundry lately, basically to allow you to customise and configure Clarionfoundry to your liking. For example, if you are interested in the ABC/Templates/Browse section, you can click the "Notify Me" button to have Clarionfoundry automatically email you a notification when an article in that section, or subsection changes. These notifications can be turned off very easily too. You are also able to ask questions, (on the main page), and be notified byemail when someone asks a question, or when someone has answered your question. You can also create your own articles. To make use of this new functionality, you need to be personally identifiable on Clarionfoundry. Therefore you now need to "subscribe" to Clarionfoundry to access the content. If your old login and password from 2 years ago does not work, simply subscribe again. You will get an email within 20 seconds of subscribing - and it's still free.

*Posted Tuesday, October 19, 2004*

## Expert Rules Beta Site

Expert Rules simplifies the time and cost of creating flexible database processing applications. Define Databases, Windows, Forms, Reports, Calculation systems defining Rules and lists of calculations; integrating to MS WORD, MS Excel, Outlook or VBA, and access all major databases systems just like you do with MS Access, VB, Clarion or any database utility. Expert Rules also supports the direct calling of industry standard Crystal Reports so users can design and see reports in any database format.

*Posted Tuesday, October 19, 2004*

## ABC Free Templates and Tools Updated Oct 4, 2004

Changes to this release of the ABC Free Templates and Tools includes a fix to the thread manager where a second browse could be opened on a second try, and a modification to the NT Service template so that services will always close after the kill event unless the ServiceRunning value is set.

*Posted Tuesday, October 19, 2004*

## Free Icons Promotion

Every Gitano Software purchase totaling $199 or more will receive the XP Collection, 140 icons in 5 formats (ICO, PNG, BMP and GIF) and 5 sizes (64x64, 48x48, 32x32, 24x24 and 16x16), for a total of 5880 individual files. The XP Collection is valued at $129.

*Posted Tuesday, October 19, 2004*

## GWBSQLAutoInc Updated

Use when you need AutoInc on the server, but need to know the value of the primary key so that child records can be added. The template is simply added to the Update Button on a form procedure. Check the included text file for the latest changes - this is a very fluid template!

*Posted Tuesday, October 19, 2004*

## BoxSoft Super Stuff 6.5 Released!

After a long wait, the new documentation and examples are finally complete. The vast majority of the templates have been enhanced, and most of the templates that were in only ABC or Clarion/Legacy are now in both (although there are still a few stragglers). NOTE: All files have been renamed to adhere to an 8.3 format, which Clarion's current 16-bit IDE handles with far fewer quirks. The installation program will delete STABMH* and STCLMH*, which now use the prefixes STAM* and STCM*. You can download all the Super Templates from www.boxsoft.net. For all purchase and upgrade issues (including passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (952) 745-4941.

*Posted Tuesday, October 19, 2004*

## JAGUAR 1.0 Beta 2

Beta 2 of the JAGUAR Java template set for Clarion has been released. New features include: Product documentation; Reporting features; Charting features; Page-loaded browses; Multi-app projects support; JDBC native drivers support; Automatic generation of

JavaDoc documents; New control template to include external Java components; Improved positioning and sizing of windows controls; MDI schema support; New JAGUAR compiler; Automatic generation of JAR files; Automatic generation of distribution folder (for easier application deployment); Support for date fields in both DATE and STRING + GROUP formats; DCT initial values support; DCT validation checks support; New options to export JAGUAR projects to third party IDEs (Eclipse & CodeGuide); New compiler options; New embed points; New Clarion statements available; New example applications; Improved JAGUAR installer. Includes the source files of both JBCL and JCSL libraries. These include all the Java classes developed to shorten the JAVA learning curve for Clarion developers, making available many of the Clarion structures (FILEs, QUEUEs, etc.) and built-in procedures (ADD, PUT, DELETE, MESSAGE, etc.) to be used in any JAGUAR application.

*Posted Tuesday, October 19, 2004*

## CWPlus ver 3.01

CWPlus ver 3.01 is now available. New in this release: A feature which allows you to restore the last cursor position in the source code and jump there when you enter again on editing in Embed Editor, Procedure Source Editor or File Editor. (thanks to Dermot Herron); Fixes for all known bugs. Price: $99 full license, $39 additional license. CWPlus is available as a full (trial) version.

*Posted Monday, October 04, 2004*

## ThinkData's 30% Off Sale Continues Through October 15

ThinkData announces a 30% off sale on all products on the ThinkData web site from September 1 to- October 15 2004, in celebration of DevCon. All of products are Clarion 5.5 and Clarion 6 and Clarion 6.1 compatible, contain both ABC and Legacy templates, and ship with complete source code. xmlFUSE, regularly $249, now $174; qbFUSE, regularly $199, now $139; OutlookFUSE, OutlookFUSE, regularly $199, now $139; EnhancedScrollClass, regularly $69, now $48.

*Posted Monday, October 04, 2004*

## Nice Touch Version 6 Upgrades

Nice Touch announces its upcoming Version 6 release of all its products with a new look and feel and requested new features. These are partially itemized on the website with a few screen shots. The first beta release is scheduled for Monday October 4, 2004 with a target Gold release at the end of October. Additional beta releases will be issued as required. Click on the Sales Page for pricing and the Wizard for the order form. The DevCon Special of 20% off has been extended until Friday October 15, 2004. Additional New Company Licenses are discounted at 50% during this DevCon Special period.

*Posted Monday, October 04, 2004*

## EasyCOM2INC 1.14

EasyCOM2INC 1.14 is now available. Changes include: New reserved words "PAGE", "PRINT", and "SCROLL"; Changed tree view trv file format for future compatibility with the class generator; Fixed incorrect method generation with "SAFEARRAY(SomeType)" parameters; Fixed incorrect method generation with string parameter by default.

*Posted Monday, October 04, 2004*

## C6.1 HotFix 9028

A hotfix release to update Clarion 6.1 Build 9027 to version 9028 is now available. If you had previously installed the 9028 alpha build which was released on September 1, you must uninstall it in order to install the production version of 9028.

*Posted Monday, October 04, 2004*

## IMDD HotFix 9028

A hotfix is now available to update IMDD Build 9027 to ver 9028.

*Posted Monday, October 04, 2004*

## Clarion Developers Challenge Week 3 Winner

First place for Week 3 of the Clarion Developers Challenge goes to Dave Sperling! Dave picked a very solid 11 out of 16 to claim first place in this week's contest! Jason Miner and Jerry Davis finished a close second with 10 correct picks. For finishing first in this week's Clarion Developers Challenge, Dave will be receiving a copy of

FotoKiss donated by Charles Edmonds and LANSRAD Software.

*Posted Monday, October 04, 2004*

## cpTracker Pro Sale

cpTracker Pro is now on sale for Clarion developers through the end of October. $69.00 per user. This is $30 off the regular price of $99 per user. Go to www.berthume.com/purchase.htm and select the Custom Solution order option and enter the sale price.

*Posted Monday, October 04, 2004*

## NET Tools Library In PDF-Tools

PDF-Tools has been updated to include the new and much improved Version 3 Netlib library, which includes tools for posting emails and files via MAPI (MS Optimised), SMTP (secure Auth now supported), HTTP and FTP. This is a free addition/upgrade and will eventually be included in all of Tracker Software's other Dev Tools (Image-XChange and TIFF-XChange).

*Posted Monday, October 04, 2004*

## Blowfish Encryption Code

J-PWBlowfish is a wrapper class for Andy Ireland's Clarion implementation of the Blowfish encryption algorithm, written by CapeSoft's Gary James.

*Posted Monday, October 04, 2004*

## LANSRAD Software Named Vendor of the Week

Charles Edmonds and LANSRAD Software have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week! Charles and LANSRAD Software are awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of FotoKISS, their widely acclaimed auction photo editor. Charles also has some very interesting tools in the pipeline that are soon to be released to the Clarion community, so stay tuned for more information.

*Posted Monday, October 04, 2004*

## EasyResizeAndSplit 2.06

EasyResizeAndSplit version 2.06 is now available. Improvements include: A new drawing mechanism; Bug fix to grab corner lines on non-MDI windows; MDI GPF bug fix; Event:Sized bug fix for non-active MDI window. This version is for Clarion 5.0, 5.5 and 6.1 (9028).

*Posted Monday, October 04, 2004*

Clarion Magazine

# DevCon 2004: Report Generation - Tricks & Tips

**by Rick Martin**

Published 2004-10-05

Diego Borojovich is an employee of SoftVelocity and the author of the new report generation classes and templates in Clarion 6.x. Unfortunately, the DevCon sessions were running really late and Diego had to squeeze his 45-minute presentation into less than 30 minutes. His was the last session of the conference and many people needed to leave to catch their flights.

First, he gave a brief review of how a report is generated. All Clarion report pages are generated WMF images. Even when no print preview is specified the report engine generates WMF images that are held in memory. A WMF file really consists of a series of Windows API drawing commands. When the WMF is displayed or printed the drawing commands are executed to render the page image.

Diego emphasized that the selected printer driver has a big effect on the generated WMF images. All of the draw commands in the WMF are in context of the current printer driver. That is why the same report can look different depending on the computer or printer generating the report.

Next, Diego talked about why it is so hard to have the same report print to text and a printer. The position coordinates of the report's controls must be translated from a large coordinate system to a small rigid one. Your report has coordinates in thousandths of inches. This lets you specify very fine placement. A text file is purely row/column. You only have about 60-70 characters across the page. An 8.5x11 report has 8500 horizontal positions in comparison. So the report generation classes translates the control positions from print X/Y coordinates to row/column text coordinates. This often leads to problems in aligning the text.

Diego strongly recommended choosing a fixed width font for reports that you will send to text. A proportional font is much harder to translate. He also recommended setting the height of your header and detail bands to a value that was evenly divisible by the fixed width font point size picked for the report. This makes the row translations much more reliable. If you don’t you can often see two different rows of the report printing on top of each other.

Unfortunately, Diego indicated that you often have to use a fair amount of trial and error to get a report to generate to both printer and text satisfactorily.

In the little remaining time, Diego briefly went over the report to XML template. He emphasized that the big advantage of using the report to XML template vs. the XML export templates, which he covered in another session, was the report XML template allows you to specify parent-child relationships in your data. The XML report template lets you specify root tags and row tags for your data. A root tag indicates a type of thing, like Product. The row tags would then be used for each product in the database. Unlike the text report generation template, the XML template requires more work setting up the controls to print and often needs additional controls that are not on the printed version of the report.

The overall feeling Diego left us with is that it takes a little effort to generate the same report to multiple types of output; especially those that convert from a proportional font to a fixed one. The PDF and HTML report outputs are much easier to use because the control position information translates much better.

## Reader Comments

Add a comment

Clarion Magazine

# DevCon 2004: Do It Yourself ABC Classes and Templates

**by Rick Martin**

Published 2004-10-05

Mike started by introducing his new SuperDiet template chain. He claims it helped him lose 40 pounds by making him eat right. He looks great.

Mike's session covered the steps required to make your own class ABC compatible and how to write a template so your class shows up in the Embed tree just like the regular ABC classes.

The first question Mike answered is, “Why write my own class and template? Isn't it easier to copy and paste the code?”
His answer was that you:

1. Reduce the amount of code you have to maintain
2. Reduce the amount of code generated
3. Easily take advantage of the logic in the other existing ABC classes.

By taking a little time to learn this technique you can simplify your life down the road by reducing your maintenance efforts.

He compared the pros and cons for manually copying the code, writing a template only solution, using a procedure and template in combination and using an ABC compliant class and template. His conclusion was that the only downside to using a class and template was people thought it was too difficult to do. So his presentation is all about showing us how easy it can really be.

Next, he covered the changes necessary to make to your class's INC file ABC compatible. Really there are only two things:

1. Add `!ABCIncludeFile` as the first line of your INC file.
2. Add `Link` and `DLL` attributes to your class definition line.

```
MyClass Class,Type,Module(‘MyClass.CLW'),Link(‘MyClass.CLW',
      _ABCLinkMode_),DLL(_ABCDllMode_)
```

There are no specific changes to the class source file (.CLW) necessary to make it ABC compatible, but Mike gave out some good pointers on designing a class. He suggested using `Construct` and `Destruct` methods to handle initializing variable and queue structures internal to the class. That way you know any dynamic memory allocation is always properly executed. He also recommended using the `DupString` procedure for copying information passed to the class from the outside. This procedure can be found in a number of the ABC class files.

Then Mike described in detail what goes into writing a template to make your new ABC compliant class generate into your procedure's Embed tree. There were five or six key step required. Mike emphasized that you don't need to understand exactly what each one is doing. Just trust that they work. All you really need to do is go through his sample template and replace his example class name with your own. That and a couple of other “search and replace” operations is all it takes. You only need to add more complex logic to the template if your class requires special initialization or other specific logic.

Overall Mike did an excellent job showing how easy it really is to make your own class ABC compatible. He also provided a small working class and template to help get folks started.

## Reader Comments

Add a comment

**Rick: Mike really made a wonderful presentation and your...**

Clarion Magazine

# DevCon2004: Metadata Management for Increased Interoperability in an Enterprise Database Environment

**by Stephen D. Gradolph**

Published 2004-10-05

Mike Gorman presented his database design and development methodology at DevCon 2004, and announced a special offer under which attendees can make use of his toolset to automate this approach. Reminding attendees how few IT projects are completed on time and within budget, Mike made a powerful case for a rigorous methodology to promote successful outcomes. His message, presented with good humor and supported by examples and detailed visuals, was well received by a large audience of DevCon attendees.

Mike began by defining the term “metadata” and providing examples. Clarion developers should recognize their data dictionaries and application files as metadata. Ideally, metadata should itself be retrievable from and maintained in a relational database. He introduced the term “metadata management” as short hand for building and maintaining the metadata database.

As background, Mike painted a scary picture of the real world of enterprise computing. Because the US Air Force relies on literally thousands of different computer systems, one study showed that 80% of their IT budget goes to develop interfaces so information could be shared. Assuming the US Federal Government suffers the same ills, Mike estimated that five billion dollars are wasted each year due to a lack of uniform data standards and interoperability among systems. Mike also shared a review of a Standish Group study, which concluded that only 16% of recent IT projects in the electrical and gas utility industry were judged successful. Breaking down the statistics, Mike was able to show that few projects fail for technological reasons alone.

Returning to theory Mike presented a complete syllabus introducing his Knowledge Worker Framework and what he called metadata models:

- Mission, function, and organization
- Information needs analysis
- Resource life cycle analysis
- Information systems and business events
- Data models

Additional charts divided data modeling into five separate layers and presented detailed diagrams of the components making up each layer. Time did not permit lengthy discussion.

Instead, as he sprinted through the presentation, Mike departed from pure methodology to give attendees more practical advice in database development projects. Mike stressed attention to the human element and appreciation of organizational inertia so that fundamental questions are wrestled with until consensus is reached. Programming should not begin until the needs of the enterprise are thoroughly understood, and no project will succeed unless the database actually meets these requirements.

Mike continued with a brief explanation of how metadata management should work in the real world. Accommodation with existing databases is often required, as well as tight integration between software development tools. In the “Reverse Engineering” scenario Mike presented, existing database schemas are extracted and imported to populate the metadata database. Then, work proceeds through the “Forward Engineering” scenario in which the new database is created automatically from the assembled metadata.

To house and manipulate the metadata, developers need what Mike called an "Upper CASE" tool, which should work closely with another "Lower CASE" or rapid application development tool, like Clarion, to translate database specifications into reality.

Finally, as time ran short, Mike gave attendees a peek at the toolset he has developed to meet this challenge: the Whitemarsh Metabase. After twenty-five years implementing his metadata management methodology on various database platforms, Mike spoke briefly about its most recent incarnation written completely in Clarion. A simple graphic offered just a few vital statistics: 125 separate database tables, fourteen separate Clarion application files, and 765 source modules, where perhaps 5% of code is written by hand. Another chart gave credit to Clarion third party development tools.

Sounds too good to good to be true or maybe ... expensive? Well, seeing is believing, and Mike announced a special offer under which the flintiest DevCon attendees could have look. Under a free membership, subscribers can download a working demo of the Whitemarsh Metabase and gain access to the complete collection of white papers, books, and other material that describe the Whitemarsh Knowledge Worker Framework. Attendees were given a contact email address.

## Reader Comments

Add a comment

**Just read the article about Mike Gorman's metadata...**

Clarion Magazine

# DevCon 2004: Threading - Just What You Need To Know

**by Nardus Swanevelder**

Published 2004-10-21

If you are still using Clarion 5.5 to develop new applications because you are scared by the new threading model then Steve Parker's DevCon presentation has the solution for you. The goals of Steve's talk were:

- Explore the essentials of pre-emptive multithreading
- Understand the concepts and vocabulary of threads
- Alleviate the "fear of threading"
- Provide a foundation on which you can base decisions about data structures
- Examine Clarion methods to deal with threads

Steve gave us the bad news first: "There is a huge amount of material available about threads and threading and almost all of it is utterly incomprehensible to normal human beings. Luckily the good news is that almost none of it is really important".

The next obvious question is: If it is so easy, what’s all the commotion about? Steve demonstrated an application developed by Carl Barnes that runs a procedure in five different threads at the same time. The procedure did a simple assignment and then tested the value after the assignment. After the example finished running the result was that only 4.5% of the 2500 assignments were successful.

What do you have to master to fix this problem?

- Few definitions
- Little knowledge
- Common sense, in other words defensive programming

Steve pointed out that all previous versions of Clarion for Windows did do real threads, but it was just hidden from you the programmer. He also pointed out that in the new threading model; each thread gets its own memory space, so a Clarion 6 application will use slightly more memory when compared to a Clarion 5.5 application.

Are you one of those people that make one or more of the following statements?

- If I’m going to use 6.x and subsequent releases, I have to deal with threads.
- Threads are difficult to deal with.
- Threads introduce all kinds of problems for me to deal with.

Steve has one word for you: *Wrong!*

Steve then looked at a couple of definitions:

*What is a process?* A process is a container for threads that you are going to create, or it is a thread that is running at the operating system level.

*What is a thread?* A thread is contained within a process, or it is a path of execution.

After explaining these definitions Steve demonstrated some other functionality in the example application. This time he disabled a "Sleep in test" function, after which the application completed all of the assignments without any errors. This function forced a thread swap which meant that the un-threaded global variable did not get the right value.

The conclusion that you can jump to after this example is that global data, global variables or global queues are the problem. Steve again had one word: *Wrong!*. The problem is not global data but static data. Static data is basically any data without the thread attribute. Remember that Global Data is by default static (not threaded) and Local Data is by default threaded (unless you tick the static attribute).

Where does static data come from? Most of us use global variables to pass data, store calculated values and Steve pointed out rightly that this is *bad*.

Do you want to know what you need to do to fix 99% of your problems?

- Thread all static data
- Eliminate static data by really passing parameters

Steve added the thread attribute to the global variable in the example application and executed the program. As Steve puts it: Hmmm… It runs just as fast, just as many cycles and no errors.

Thread your global data, and you just solved 99% of errors you are going to have.

The next step is to use the new template features to pass parameters. The new features even have an expanded view button that enables you to see everything that you type.

But what if you have a gazillion global variables spread across a bunch of EXEs and even more DLLlls, some are in the Dictionary, some are in various Application’s data area, some are in data embeds, etc. As Steve pointed out, programmers may say "I have more places I need parameters than a politician has excuses." And Steve’s reply is: "In other words: I am just too lazy."

Everything worked fine in previous versions so why can’t I just use the "old" model?

## You asked for it!

Try the new Global Cooperative Thread template. This will take you back to the cooperative thread model but you will still have a separate memory space per thread.

What happens to the example application if the new template is added to it? It completes the threads but is very slow (900 compared to 2500). This is not a big problem; in a real life application the chance of running more than two threads at the same time is slim.

If everything else failed and you have to use static data you can protect yourself in the pre-emptive threading environment by using synchronization objects.

## Synchronization objects

What are synchronization objects? Synchronizing threads causes multiple threads to have identical data, or synchronizing works by restricting the access to data to one thread at a time.

Some synchronization objects are waitable and some are not...

Waitable means that it is possible to test or to see whether the target object is already locked or not. In Clarion class wrappers this corresponds to a .TryWait() method.

If Pierre has the object locked and Bob wants it, Bob has to wait. Funny enough, if the wait succeeds it is a good thing; it means that you did not have to wait.

So what synchronization objects does Clarion 6.x provide?

- Critical sections

- Mutexes
- Semaphores
- Read/Write Locks

Clarion also has a critical procedure, but this is not a sync object per se. It is a class wrapper around a critical section.

Of these you need to understand and be able to use the critical section and the mutex. These two will fix 99% of your remaining problems. See Steve's article "Implementing a Critical Section: Fast and Effective" in this issue.

## Mutexes

According to the Clarion documentation a mutex is used when you want to allow one thread to access a resource. A mutex can synchronize between processes or between applications. A mutex can be named and is running at the operating system kernel and therefore is "waitable". It is not as easy as a critical section but it offers more features.

A mutex should be named. The name must, must, must be unique!

A mutex is waitable. You have the choice of waiting until the mutex is released or you can give the user a message or you can bail out. You have the choice to terminate the wait.

The operating system will kill a mutex if it was not used for a while.

If you using a mutex on a procedure remember to make the mutex threaded:

That's it. After this Steve demonstrated how to use a mutex to limit an application to a single instance on a machine.

The following questions were asked:

> **Q:** How do you prevent two different vendors from using

the same mutex name?

**A:** Use words no one else knows. Or use a combination of your application’s path and application’s name as the mutex name.

**Q:** What do you mean by a global lookup?

**A:** People populate data to a global queue and then do a lookup against that queue. Bob Z said that you can use the new In-Memory Driver (IMD) for this functionality, as the IMD is using the File Manager class which is threaded and therefore is thread-safe.

**Q:** If you write to a global variable when you open your application and after that you only read from that variable, will this be a problem in the new threading model?

**A:** As long as you are not going to make a change to this variable this is safe to use.

**Q:** How does the new global cooperative thread compares speed wise if compared with a critical section?

**A:** The new global cooperative thread model will be slower than a critical section but in a real life application this should not be noticeable to the end user.

## Summary

Steve showed that moving your Clarion 5.5 applications to Clarion 6 is not that difficult after all. You should try to add the thread attribute to most of your static data. If it is not possible to add the thread attribute to your static data you should use a critical section or a mutex to protect that data. If you are too "lazy" to use these for your global queues you can use the In-Memory-Driver which is thread-safe by nature.

## For more information

### Articles by Steve Parker:

- A Naïve Look At Pre-Emption
- A Naïve Look at The Mutex
- Mutexes: Serializing File Access

### Articles by Dave Harms and Carl Barnes:

- Demystifying C6 Threading (Part 1)
- Demystifying C6 Threading (Part 2)
- Demystifying C6 Threading (Part 3)

### Article by Dermot Herron

- Converting Clarion 5.5 Apps To Clarion 6.1

### Documents from Clarion

- Multi Threaded Programming
- Pre-emptive and Cooperative Thread models
- Cooperative threading in a pre-emptive environment

---

*Nardus Swanevelder was born and raised in South Africa. He was a networking engineer for seven years before he moved over to the commercial side of the business. Nardus has developed a Sale Cycle Management system for the Information and Communication Technology industry. He has been programming in Clarion since 1989, and holds B.Com and MBA degrees. In his spare time Nardus lectures Financial Management to B. Com Hons students at North-West University.*

## Reader Comments

Add a comment

**I forgot to say thanks to Steve for an excellent...**
**Article: "The operating system will kill a mutex if it was...**
**Thanks, a bit more technical but exactly what I wanted to...**

Clarion Magazine

# Implementing a Critical Section: Fast and Effective

**by Steven Parker**

Published 2004-10-21

The Critical Section is the construct most commonly used to protect reading and writing non-threaded, static data in multi-threaded applications. There are two ways to implement a Critical Section:

- You can study the Windows API and use the API directly
- You can use the synchronization classes/wrappers provided by SoftVelocity

The latter is much, much easier. In fact, using the synchronization classes from SoftVelocity, a Critical Section can be implemented in just four lines of code.

## But first, a word from our sponsor

Why do you need to know about and use Critical Sections?

Suppose you have a state/province lookup file. When you start your program, you open this file and since it's used everywhere, it isn't threaded. Further suppose that you store the state/province abbreviation in the file but display the full name to the user.

Now, in one thread you are adding a new record and have to look up the state/province. In another, you are editing an existing record. Both procedures will do something like:

```
STA:Code = CUS:State
Access:STATES.Fetch(STA:CodeKey)
!lookup procedure call
CUS:State = STA:CODE
```

What prevents Windows from switching to another thread in your application, between the .Fetch and the assignment? Or between priming the key and doing the fetch? Answer: "nothing." And now you have either the wrong value stored in the file or the wrong value displayed on the form.

(Okay, let’s be honest: the odds on this happening are fairly slim. A user could not, in all likelihood, switch from one procedure to another fast enough to mash this lookup. A couple of reports or processes - or hand coded tight loops- running simultaneously *might*. Still, the odds *are* slim. But, as they say, it only takes once ....)

You also have some very irate support calls coming.

The same sort of thing happens with global queues:

```
que:Value = primed:Value
Get(MyQueue,+que:Value)
loc:OtherField = que:OtherField
```

If two procedures are trying to get values from the queue and there is the possibility of a thread switch before all three lines above have completed, you *cannot* rely on the value in `loc:OtherField`. Read that sentence again, until you are certain you understand what’s at stake.

In general, any time you try to access a non-threaded data object (file, queue, view, etc.) where there is no built in concurrency checking, you need to be concerned about a thread switch and unreliable values. (It is very nice that concurrency checking is built into the Form template, isn’t it?)

Wrapping the reading and writing of a non-threaded data object in a Critical Section solves this problem. A Critical Section locks the object, preventing any other thread from accessing the data at all.

## A Critical Section in Four Lines of Code

Using SoftVelocity’s implementation (see *Multi-Threaded Programming Guide* on your installation CD), it only takes four lines of code to create and use a Critical Section.

In Global Data (I use the After File Declarations embed):

```
include('CWSYNCHC.INC'),ONCE
```

This includes the class definitions for Critical Sections (it also includes definitions for some other synchronization objects: Critical Procedures, Mutexes and Semaphores). SoftVelocity’s wrapper for the API code is found in the associated .CLW file.

In the same embed:

```
MyCriticalSection CriticalSection
```

instantiates (creates) a Critical Section. `MyCriticalSection` is a useable object.

Two down, two to go.

Immediately before you need to access (read "lock") a non-threaded data object, open ("enter") the Critical Section:

```
MyCriticalSection.Wait()
```

When you have read/written and it is okay for another thread to access the data:

```
MyCriticalSection.Release()
```

That’s it.

The final code for accessing a non-threaded, static, datum then ends up looking like this:

```
MyCriticalSection.Wait()
STA:Code = CUS:State
Access:STATES.Fetch(STA:CodeKey)
!lookup procedure call
CUS:State = STA:CODE
MyCriticalSection.Release()
```

or

```
MyCriticalSection.Wait()
que:Value = primed:Value
Get(MyQueue,+que:Value)
loc:OtherField = que:OtherField
MyCriticalSection.Release()
```

## Watch Out!

There is only one caveat, one thing you *must* be careful about. If you do not release the Critical Section at the earliest possible moment, your application will appear to hang. Whenever another procedure tries to enter the Critical Section, if the previous one has not been released, it will be blocked until such time as the first procedure releases the critical section.

The demonstration .APP does just this; it enters a Critical Section when opening a browse and releases it only in the `Kill` method. Open the browse (select Browse | Browse with Critical Section from the main menu). Open another instance. Even though the browse is `START`ed, only one instance will open. Any other instances will open, one at a time, when you close the visible instance.

This is not to imply that you can never use a Critical Section in a template procedure. You could, if you wanted, wrap the Parent call in `ThisWindow.TakeCompleted` in a Critical Section (only the one method call would be within the scope of the Critical Section). I am saying that you must have a synchronization object open for as short a time, as few lines as possible.

Other than that, it's really quite easy.

Download the source

---

*Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.*

## Reader Comments

Add a comment

**Steve is a great guy, but since he has been using Clarion...**
**Ah, yes, I can see it! 32 bit IDE,...**
**Steve, On a multi dll app, do all apps need critical...**
**Because a CS is implemented as an object, it would be...**

Clarion Magazine

# Using Clarion's Business Rules, Part 1

**by Nardus Swanevelder**

Published 2004-10-21

I am sure the following scenario is not new to you. Your client phones you and asks you to change his application so that it is not possible to complete a screen if the user hasn't entered data in some field, such as the zip code. You can do one of the following:

1. Go through every procedure in the application and add code to the OK button that checks the zip code field for a value. If you read Bruce Johnson's ABC book you will add the code to the `TakeCompleted` event. Recompile the app and send it to the client.
2. Go through each table in the dictionary and add the `Require` attribute to the zip code fields. Go into the application, synchronize with the dictionary, recompile and send it to the client.
3. Using Clarion's Business Rules, go into the Global properties of the application, add the rule as a global rule, recompile the application and send it to your client.
4. Add the rule to a table (TPS for example), email it to the client, and that is it.

In the first part of this article I will look at what you need to do to implement the standard business rule functionality into your application. In the second part I will show you how to write a new template that will enable you to change your rules at runtime.

## What is a business rule?

First things first. What is a business rule? According to the Clarion documentation, "business rules are precise statements that describe, constrain, and control selected data elements within your application."

## How do the business rules work?

Business rules are handled in C6 by classes, which are wrapped up in templates. This wrapping consists of a global extension template and a procedure extension template. The global extension allows you to define rules that can be used throughout your application. In each procedure, you can choose to override all, or part, of these business rules.

You can find the Business rules files in the Clarion\LIBSRC folder. They are:

- ABRULE.INC Business rules class declarations
- ABRULE.CLW Business rules class methods

You can find the Business rules template file in the Clarion\template folder. It is:

- RULESMAN.TPL Business rules declarations

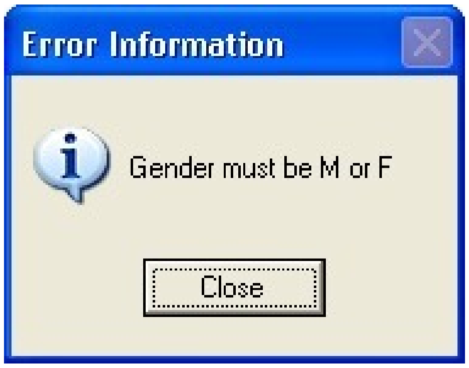
## An Example

The following example shows the necessary steps to use standard Clarion template prompts to implement business rules in an application.

### Step 1 – Add the global rules

The template wrapper makes adding global rules a breeze. Just follow these easy steps:

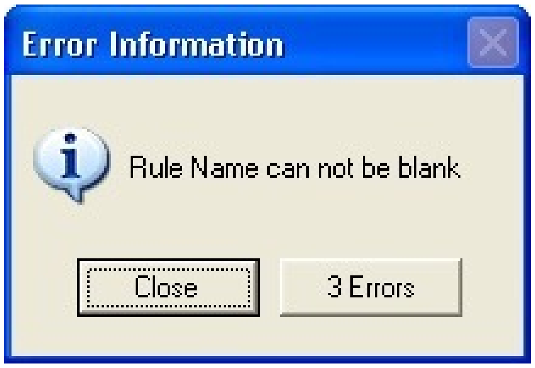
1. Open your app and go to the global settings.
2. Click on the Extensions button, and then click on the Insert button.
3. Scroll down until you get to SVBusinessRules, then click on Global Business Rules Manager.
4. Supply a Rules File Name. The template stores the rules in this file. The file will have a .brf extension but it is a plain text file.
5. You can leave all the other prompts as the default values are sufficient for now
6. Click on the Global Rules button, and then click on the Insert Button. Type in a global rule name and description. For example: `MyGlobalRules` and `My Global Rules`.
7. Click on the Insert button. Complete the Rule Name; use `Gender` as an example. Complete the Rule Description using `Gender must be M or F`. Remember that this description will be displayed when the rule is broken. Complete the Evaluate Expression, using `PEO:Gender = 'M' or PEO:Gender = 'F'`. Keep in mind that if this expression evaluates to zero it means that the rule is broken. If the expression you type is incorrect you will not get an error at compile time or at runtime. The expression will always evaluate to 1 and the rule will never be broken. Complete the Field to link to, use `PEO:Gender`. Remember that it is *Field* to link to and not *Control* to link to, so do not use `?PEO:Gender`.
8. Click on the Controls button. The next screen has two tabs, the first tab says Enable/Disable and the second tab says Hide/UnHide. If you want controls to be enabled or disabled if the rule is broken use the first tab otherwise use the second tab if you want the controls to be hidden or unhidden. If you want the OK, Cancel and related buttons click on the Add OK and Cancel Controls button.
9. Click on OK until you get back into your app.
10. Compile and run the app. If you are using the app included in the source of this article, click on the Browse menu option, then click on Browse the people file, and enter on any an item in the browse. On the update form change the gender to something other than M or F. A small button appears next to the Gender control, this indicates to the user that a rule is broken. If the user clicks on this button this fault message in Figure 1 is displayed.


Error Information

Gender must be M or F

**Figure 1. Single rule error message**

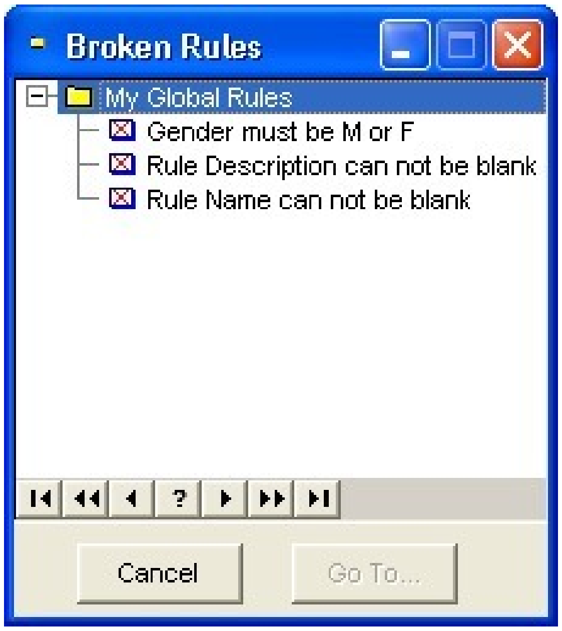
11. If more than one rule is broken the screen will look like Figure 2.


Error Information

Rule Name can not be blank

**Figure 2. Breaking multiple rules**

12. When the user click on the 3 Errors button, a window like that in Figure 3 appears. Clicking on the Go To button directs the user to the specified control.


Broken Rules

My Global Rules
- Gender must be M or F
- Rule Description can not be blank
- Rule Name can not be blank

**Figure 3. Broken rules detail**

That was fairly easy. But let's say that you want to enforce a rule on every procedure except for one specific procedure, how do you do that?

1. Open the procedure where you want to override the global rule. Click on Extensions, choose Local Business Role Manager. If a global rule is active on this procedure it will be listed under the Used Global Rules Button.
2. Click on the Used Global Rules Button. Choose the rule you want to override and click on Properties. If you want to disable the rule just click on the Disable this Rule tick box. You can change a couple of properties but you can not change the evaluate expression. If you want to change the evaluate expression you will have to disable the global rule and add a local rule. Click on OK and OK. Next, add a local rule.
3. When adding a local rule, click on the Local Rules button and then click on the Insert button. Give the Rule a name and a description. Use `MyLocalRule` and `My Local Rule` as an example.
4. Click on the Insert button and complete the Rule Name and Rule Description. Use `FirstName` and `First Name can not be blank` as an example. Type an evaluation expression like `PEO:FirstName <> ''`. At Display to RHS of, select `?PEO:FirstName`. This together with the Distance from RHS setting tells the template where to display the rule-is-broken indicator button.
5. Click on the Controls button. The next screen has two tabs: the first tab says Enable/Disable, and the second Hide/UnHide. If you want controls to be enabled or disabled if the rule is broken use the first tab, otherwise use the second tab if you want the controls to be hidden or unhidden. If you want the OK, Cancel and related buttons, click on the Add OK and Cancel Controls button.
6. Remember, if you use local rules you have to add all controls as Hot Fields. That means that if you use other controls or procedures in the evaluate string they need to bound as well. To add the Hot Fields click on the Hot Fields button and then click on the Insert Button. Select the control you need (`PEO:FirstName`) and click on OK until you are back in the app.
7. Compile and run the app. Select the Browse Menu, Browse the People File option and insert a new record. You should see a rule-is-broken indicator next to the First name and Gender.

## Multi DLL

The next question that needs to be answered is what happens with Multi DLL apps. The Global Extension has a Mode section with two options: Original and Clone. (And you thought cloning was only for the medical profession) So how do you this? Add the Global extension to your data dll and set the Mode option to Original. Add all the rules in this app and add the Global extension to all the other dlls and the exe but change the Mode option to Clone.

## Summary

This article is not supposed to teach you everything about Business Rules but I hope it gives you enough information to start using it in your applications. If you need more information you can refer to the following references:

- Clarion Documentation: Template User Guide – Business Rules Templates
- Clarion Help: Templates by Topic – Business Rules
- Application Guide – Clarion Business Rules Manager (available from *SoftVelocity)*

What have you learned so far? You know how to add Global rules to an application, how to override it on a local procedure as well as how to add a local rule. What does this give you?

- Centralizes business rules definitions
- Apply these to fields populated anywhere in the application
- Can Enable/Disable/Hide/Unhide controls based on whether rules are broken or not

The problem I have with the above is that if you make changes to the rules you have to recompile the app before the changes become active. In the next article I will look at creating a template that will allow you to change the rules at runtime.

Download the source

---

*Nardus Swanevelder was born and raised in South Africa. He was a networking engineer for seven years before he moved over to the commercial side of the business. Nardus has developed a Sale Cycle Management system for the Information and Communication Technology industry. He has been programming in Clarion since 1989, and holds B.Com and MBA degrees. In his spare time Nardus lectures Financial Management to B. Com Hons students at North-West University.*

## Reader Comments

Add a comment

Clarion Magazine

# Using Clarion's Business Rules, Part 2

by Nardus Swanevelder

Published 2004-10-29

In the previous article in this series I showed how to add Global business rules to an application, how to override a business rule on a local procedure and how to add a business rule to a specific procedure. As I indicated in that article I would like to be able to change the rules at runtime and that is what I am going to look at now.

In order to change the rules at runtime you need to understand more of the inner workings of the RulesManager class and template. Let's look at what code the template adds to your procedure.

## Global Declarations

The Global extension adds the following code:

```
Include('AbRule.inc'),once
GlobalRule long,thread
```

Instantiate local and global class as well as Rules:

```
BusinessRulesManager      RulesManager
MyGlobalRules      RulesCollection
MyLocalRule      RulesCollection
```

## In the Init embed

Assign broken rule icon and rule description for global rules:

```
MyGlobalRules.SetErrorImage('~BRuleNo.ico')
MyGlobalRules.SetDescription('My Global Rules')
```

Assign broken rule icon and rule description for local rules:

```
MyLocalRule.SetErrorImage('~BRuleNo.ico')
MyLocalRule.SetDescription('My Local Rules')
```

Start adding the global rules applicable to this procedure:

```
MyGlobalRules.AddRule('Gender','Gender must be M or F',|
 'PEO:Gender = ''F'' or PEO:Gender = ''M''',?PEO:Gender,3)
```

Start adding local rules applicable to this procedure:

```
MyLocalRule.AddRule('FirstName','First Name can not be blank',|
  'PEO:FirstName <<> ''''',?PEO:FirstName,3)
```

Add the controls that belong to the global rule. The rule action is set to Disable, and as the following code shows, if the rule is broken, the OK and Cancel buttons will be disabled:

```
MyGlobalRules.AddControlToRule('Gender',?Ok,RuleAction:Disable)
MyGlobalRules.AddControlToRule('Gender',?Cancel,RuleAction:Disable)
```

Add the controls that belong to the local rule. The rule action is set to Disable. If the rule is broken, only the OK button will be disabled.

```
MyLocalRule.AddControlToRule('FirstName',?OK,RuleAction:Disable)
```

Set icons (global) for error display if more than one rule is broken:

```
MyGlobalRules.SetEnumerateIcons('~BRules.ico','~BRuleOk.ico','~BRuleNo.ico')
BusinessRulesManager.AddRulesCollection(MyGlobalRules)
```

Set icons (local) for error display if more than one rule is broken:

```
MyLocalRule.SetEnumerateIcons('~BRules.ico','~BRuleOk.ico','~BRuleNo.ico')
BusinessRulesManager.AddRulesCollection(MyLocalRule)
```

Set icons (Rules Manager) for error display if more than one rule is broken:

```
BusinessRulesManager.SetEnumerateIcons('~BRules.ico','~BRuleOk.ico','~BRuleNo.ico')
BusinessRulesManager.SetGlobalRuleReferences(GlobalRule)
```

This code will be added if you tick "Check Rules After Open Window" box in the Local extension template:

```
BusinessRulesManager.CheckAllRules(1)
```

## ThisWindow.TakeAccepted PROCEDURE

The RulesManager traps the Accepted event to determine if the user clicked on the error-indicator:

```
BusinessRulesManager.TakeAccepted(Accepted())
```

Check all Rules:

```
BusinessRulesManager.CheckAllRules(True)
```

## ThisWindow.TakeCompleted PROCEDURE

Call CheckAllRules with a parameter of 0 which means that a broken rule indicator will not be displayed if a broken rule is found. If a broken rule is found Return Level:Notify..

```
IF BusinessRulesManager.CheckAllRules(0)
  RETURN Level:Notify
END
```

## ThisWindow.TakeNewSelection PROCEDURE

Call CheckAllRules and display any errors:

```
BusinessRulesManager.CheckAllRules(True)
```

Remember that’s all code that’s generated for you. So why did you have to see it?

## Methods

If you look carefully you should see that you need only two methods to add your own rules: MyGlobalRules.AddRule and MyGlobalRules.AddControlToRule. And if you set up a couple of tables to contain these rules, you can create a truly dynamic system of business rules, where changing validation is as easy as updating the table data.

AddRule is declared in the AbRule.Inc file as follows:

```
AddRule PROCEDURE(STRING RuleName,STRING RuleDescription, STRING RuleExpression,<LONG
  ControlNum>,LONG OSR=3)
```

As you can see you need a RuleName of type STRING, a RuleDescription of type

STRING, a RuleExpression of type STRING, a ControlNumber of type LONG and an Offset of type LONG.

What should the length of these string variables be? Look at the definition of the prompts in the template:

```
%RuleName = @S255
%RuleDescription = @S255
%RuleDefinition (Evaluate expression) = @S255
```

One problem is that you would enter the control as a string such as ?PEO:Gender but the class requires a control number as a long. I will get back to this later.

Now look at the AddControlToRule procedure, declared as follows:

```
AddControlToRule PROCEDURE(STRING RuleName,|
                           UNSIGNED pControlFeq,BYTE pAction)
```

This method accepts need a RuleName of type STRING, a ControlNumber of type UNSIGNED, and an Action of type BYTE.

Remember that the RuleName is declared as @S255; you enter the control as a string, but the class requires it, this time, as an unsigned integer. Again, I'll come back to that.

What actions can you send to the method? This time look at the AbRule.Inc file. At the top of the file you will find the following:

```
RuleAction:None     _EQUATE(0)
RuleAction:Hide     _EQUATE(1)
RuleAction:UnHide   _EQUATE(2)
RuleAction:Disable  _EQUATE(3)
RuleAction:Enable   _EQUATE(4)
```

Okay, so now you know what methods to call and you know what the properties are. At the beginning I said that it is possible to store the rules in a table like a Topspeed table. Here are the fields in the table:

```
Rules
RuleName            String(255)
RuleDescription     String(255)
RuleExpression      String(255)
RuleControl         String(255)
RuleOffset          Byte

ControlsPerRule
RuleName            String(255)
RuleControl         String(255)
```

```
RuleAction          Byte
```

Each table also needs a key based on the `RuleName`, as well as a one-to-many relationship declared in the dictionary between the two tables, based on the previously created keys. See the attached code for an example.

Why two tables? You need to be able to add more than one control to a rule, keeping in mind that these controls are the ones that will be disabled/hidden if the rule is broken.

Next, add browse and update procedure to your application. This will update the `Rules` and `ControlsPerRule` tables.

Okay, you have tables and you can update the rules. Now how do you apply it?

In the next article I will demonstrate how you apply the two methods to enable you to change your rules at runtime, including how to deal with the problem of translating to a control number at runtime.

Download the source

---

*Nardus Swanevelder was born and raised in South Africa. He was a networking engineer for seven years before he moved over to the commercial side of the business. Nardus has developed a Sale Cycle Management system for the Information and Communication Technology industry. He has been programming in Clarion since 1989, and holds B.Com and MBA degrees. In his spare time Nardus lectures Financial Management to B. Com Hons students at North-West University.*

## Reader Comments

Add a comment

Clarion Magazine

# Critical Procedures: Synchronization for the Lazy

**by Steven Parker**

Published 2004-10-29

I recently wrote about the most commonly used thread synchronization object, the Critical Section, using the classes supplied with Clarion 6. In that article, I showed two things I think are very important:

- Implementing a Critical Section, using SoftVelocity's classes, is *very, very easy*

and

- Making a mistake, causing your application to hang, is *very, very easy*

The first point: Using SoftVelocity's classes, you can implement a Critical Section in just four lines of code.

The second point: Using a Critical Section, either via SoftVelocity's classes or using the APIs directly, and failing to call its Release method it in a timely manner (whether through a typo, putting the call to Release in the wrong place or sheer forgetfulness), locks your application the next time any procedure using the Critical Section is called.

## Enter the Critical Procedure

Almost as if the SoftVelocity development team knew I'd forget to release my Critical Sections, they created the Critical Procedure. A Critical Procedure is actually a class. A CriticalProcedure object manages a Critical Section for me. I like things that do for me.

A Critical Procedure can protect all or part of a procedure, but it doesn't, on its own, know when I want to start protecting an operation, neither can it determine what Critical Section I want to use. (Remember, a Critical Section is at the heart of a Critical Procedure.) So I do have to tell it these two things: what object I want to use, and when to enter the Critical Section. But, using a Critical Procedure, I use ABC standard syntax:

```
CritProc.Init(<name of my synchronization object>)
```

So, at least, I don't have to "Wait" or "TryWait." "Init" is something I understand.

What a Critical Procedure can do, without me, is release (terminate) the Critical Section.

In other words, a Critical Procedure not only allows me to forget to release the Critical Section, it expects me to. Okay, since it has no callable Release method so it is fairer to say it understands that I used a Critical Procedure instead of a Critical Section because I know I can't be trusted to remember. A class that knows its user is untrustworthy ... I like that.

SoftVelocity's doc, "Multi-Threaded Programming Guide," characterizes a Critical Procedure as follows:

> The CriticalProcedure class is a very easy way to use an ISyncObject interface.
>
> If you create a local instance of a CriticalProcedure and initialize it, then it will look after the waiting for a lock and

> releasing the lock on the ISyncObject for you. The main advantage of using the CriticalProcedure class to handle the locking and releasing for you is that if you have multiple RETURN statements in your procedure, you do not have to worry about releasing the lock before each one. The destructor of the CriticalProcedure will handle that for you.

It strikes me that because the Release is done for me, a Critical Procedure is much more template-procedure friendly. I can sprinkle Critical Procedures liberally in my Form procedures. I can use a Critical Procedure in TakeCompleted, Before Parent call all over the place just for the warm, fuzzy feeling I get from practicing safe computing. And I can take satisfaction from not having to close what I've opened. Someone else gets to clean up after me.

However, it is the third sentence that's really important: "The main advantage of using the CriticalProcedure class to handle the locking and releasing for you is that if you have multiple RETURN statements in your procedure ...." This means that no matter how many Return statements I have in my procedure, all the exit points are covered.

Consider the function I presented in "List Box Marking" to determine whether or not a particular record is tagged:

```
IsTagged(Long),Byte
IsTagged    Procedure(pLong)
   CODE
   TAG:Ptr = pLong
   If Access:TagFile.Fetch(TAG:PtrKey)
     Return(False)
   Else
     Return(True)
   End
```

Here is the same code converted to a class method ("A Class for Tagging"):

```
shpTagClass.IsTagged     Procedure(String pPoint)!,Byte
   CODE
   Self.TagQueue.Ptr = pPoint
   Get(Self.TagQueue,Self.TagQueue.Ptr)
   If ~ErrorCode()
     Return(True)
   Else
     Return(False)
   End
```

(Note: similar code is used by many of us to limit procedures to a single instance.)

This code is not thread safe. Two reports or processes could be calling IsTagged simultaneously, causing incorrect results if there is a thread switch between calling IsTagged and getting the result. Second, there are two exit points.

Normally, I would make IsTagged thread safe by wrapping the call to it in a Critical Section:

```
ThisCritSec.Wait()
  If IsTagged(value)
    !Do things
  End
ThisCritSec.Release()
```

But, if this procedure were more complex, I would need to implement my Critical Section inside the procedure. This means I have to be really careful about doing my Release calls. So, IsTagged gives you the idea.

How much more complex would a procedure have to get? How about reading a master queue and copying it to a new queue on initializing a new thread?

With a Critical Procedure, initializing the Critical Procedure is *all* that is required. A Critical Procedure's destructor ensures that Release *is* called.

## Implementing a Critical Procedure

Oddly, while the CriticalProcedure class makes managing synchronization easier, implementing it is harder. Where implementing a Critical Section took just four lines of code, a Critical Procedure require substantially more.

Six.

That's a full 50% increase.

After File Declarations:

```
Include('CWSYNCHC.INC'),ONCE
Include('CWSYNCHM.INC'),ONCE
```

These lines include all the necessary underlying classes and APIs.

After Global Includes:

```
ChestOfDrawers &ICriticalSection
```

creates a reference to SoftVelocity's ICriticalSection class. (Note: there isn't anything magical about what you name your Critical Section object.) So all further instances of the ChestOfDrawers object will refer back to the same object.

I add this code in Program Setup:

```
ChestOfDrawers &= NewCriticalSection()
```

This sets up a reference to a Critical Section instance. And, finally, I create a Critical Procedure:

Local Data (in the procedure where the Critical Procedure will be used):

```
CritProc CriticalProcedure
```

Then in Procedure Init or just before where I need protection to begin:

```
CritProc.Init(ChestOfDrawers)
```

Total: six lines.

## The Example Programs

The first example uses the ubiquitous People.APP. This app show a simple implementation of a Critical Procedure. It also shows how not to use it (just as I used it to show how not to use a Critical Section).

The Critical Procedure is initialized on entering a browse. It will not be released until an exit point is hit. In a browse, there is only one – when the browse closes. So the Critical Procedure has a life span matching that of the browse.

Start several instance of the browse. Notice that the thread number is displayed on the screen. But, if you started more than one, you will see only one. No matter how you move it around the screen, attempting to unhide the other instances, you will see only one.

Now close the browse. Bam! Here comes another one and it displays a different thread number. It has been just sitting there waiting to get the Critical Section. Blocked, exactly as documented. (And not very user friendly – *don't* use this as a way to limit browse instances.)

The second example uses a Source procedure and features multiple exit points:

```
Count = 0
ThisThread = Thread()
Open(Window)
0{Prop:Text} = 'Thread: ' & ThisThread
CritProc.Init(ChestOfDrawers)
Accept
  Loop Count = 1 to 100000
    Display(?Count)
    If ThisThread % 2 = 0 and Count > 50000
      Return
    End
  End
  Return
End
```

If you wish, you can copy this procedure and remove the first return. You can also change the Accept to:

```
Display
Loop
```

to eliminate any features built into Accept.

If you start two instances quickly, one will execute completely, then the other will pop up and begin. In other words, Critical Procedures are well adapted to Source procedures (where they're most likely to be needed).

While automatically taking care of calling Release is a very handy thing, the Critical Procedure is *especially* useful if you need protection inside procedures with multiple exit points.

Download the source

---

*Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.*

## Reader Comments

Add a comment

Clarion Magazine

# Extending the FileManager Class: Shorthand Fetches

**by Mike Hanson**

Published 2004-10-29

One of the most commonly used commands in Clarion is GET, which can retrieve records from both files and queues. Ever since the early days of Clarion for DOS, Clarion developers have primed key fields, executed GET commands, and checked for errors. Then the ABC classes introduced the new Fetch method, to be used instead of GET. Surprisingly, this new approach was not much better. However, it can be simplified and streamlined quite easily.

Using the legacy approach, this kind of code is exceedingly commonplace:

```
My:ID = SomeID
GET(MyFile, My:IDKey)
IF ERRORCODE()
  !Handle Error
END!IF
```

Of course, the more key field components there are, the bigger this chunk of code will be:

```
My:CusID = SomeCusID
My:OrdID = SomeOrdID
My:Row   = SomeRow
GET(MyFile, My:CusOrdKey)
IF ERRORCODE()
  !Handle Error
END!IF
```

I should stress that the GET command is generally used only when the key value is unique. Otherwise, you cannot predict which record will be retrieved. If the key value is not unique, then you should use SET+NEXT instead. (Actually, using a VIEW is a much better solution, which I will cover in a future article.)

In the ABC classes, the FileManager class implements a couple of methods called Fetch and TryFetch, which do the equivalent of the GET command (with some error checking and other stuff thrown in). The code to use these new methods does not look much different than the traditional GET code:

```
My:ID = SomeID
IF Access:MyFile.Fetch(My:IDKey)
  !Handle Error
END!IF
```

Ironically, ABC actually requires *more* typing and contains funny punctuation characters too! I think a golden opportunity was missed to simplify this code. Instead, this approach should be supported:

```
IF Access:MyFile.Fetch(My:IDKey, SomeID)
  !Handle Error
END!IF
```

This code is more concise, is just as lucid, saves typing time, offers an opportunity for tighter error checking, and eliminates "noise" in your source code. All of these are laudable goals.

Fortunately, Clarion does permit a better solution. Simply go into any Application's global Classes tab and change the default FileManager class to something else. The custom alternative must do everything the regular FileManager does, which is accomplished by deriving and extending the existing FileManager class.

## MyFile.inc

The first step is to create an INC file with the class declaration:

```
!ABCIncludeFile

OMIT('_EndOfInclude_', _MyFileManagerPresent_)  ! Omit this if already compiled
_MyFileManagerPresent_ EQUATE(1)

  INCLUDE('ABFILE.INC'),ONCE

MyFileManager CLASS(FileManager),TYPE,MODULE('MyFile.clw'), |
                LINK('MyFile.clw',_ABCLinkMode_),DLL(_ABCDllMode_)
Fetch           PROCEDURE(KEY K,? F1,<? F2>,<? F3>,<? F4>,<? F5>,|
                   <? F6>,<? F7>,<? F8>,<? F9>),BYTE,PROC,VIRTUAL
TryFetch        PROCEDURE(KEY K,? F1,<? F2>,<? F3>,<? F4>,<? F5>,|
                   <? F6>,<? F7>,<? F8>,<? F9>),BYTE,PROC,VIRTUAL
AssignKeyFields PROCEDURE(KEY K,<? F1>,<? F2>,<? F3>,<? F4>,<? F5>,|
                   <? F6>,<? F7>,<? F8>,<? F9>),BYTE,PROTECTED
              END!CLASS
_EndOfInclude_
```

The important things to note here are:

**`_ABCIncludeFile_`** - This is a "flag" to tell application generator than the INC file is ABC compatible. It *must* be the first thing in the file. (Remember to place this INC file into Clarion LIBSRC directory, or it will not be scanned, recognized and used.)

**`INCLUDE('ABFILE.INC')`** - Now the INC file knows everything about the FileManager class declaration.

**`CLASS(FileManager)`** - The new class is derived from the FileManager class.

**`MODULE('MyFile.clw')`** - This is where the class methods will be defined.

**`LINK('MyFile.clw',_ABCLinkMode_),DLL(_ABCDllMode_)`** - Should the source be compiled and linked into the current APP, or is it present in an external DLL? The _ABCLinkMode_ and _ABCDllMode_ compiler equates are maintained automatically by the AppGen, and they will usually have values of 1 and 0, or 0 and 1.

`Fetch` and `TryFetch` have rather strange parameter prototypes:

```
(KEY K,? F1,<? F2>,<? F3>,<? F4>,<? F5>,<? F6>,<?F7>,<? F8>,<? F9>)
```

Both take a `KEY`, just like the regular `Fetch` and `TryFetch` methods. However, they also take *at least one* key component value, and there may be as many as nine. In reality, you probably would not use `Fetch` or `TryFetch` if the key had nine components, because the resulting procedure call would be no cleaner than using the old approach. Line continuation characters and indentation oddities would probably obfuscate it, whereas using original individual field assignments could clarify the nature of the key component values being used.

As you can see, the derived class uses the same method names as the base class. If the parameter lists were identical, then this class would be overriding (i.e. replacing) the original method. In this case, however, the extra parameters indicate that it is an overloaded method. In other words, there are now two methods called Fetch and two called TryFetch. Depending on the actual parameters passed, one or the other will be used by the compiler.

It is partially for this reason that the first F1 parameter is required, while the rest are optional. Otherwise, the compiler would be confused whenever it encountered a Fetch call without any field components – it wouldn't know if it should use the original one in FileManager, or the overloaded one in MyFileManager. Of course, the new method is pointless without passing at least one key component value, so F1 is logically a required parameter anyway.

The prototype for AssignKeyFields is slightly different:

```
(KEY K,<? F1>,<? F2>,<? F3>,<? F4>,<? F5>,<?F6>,<? F7>,<? F8>,<? F9>),|
   BYTE,PROTECTED
```

Unlike Fetch and TryFetch, the parameter for the first key component is optional. It does not affect this scenario, but will come into play for other needs. For now, it is known that the caller will pass in at least one value.

Also note that the method is PROTECTED. This means that it can be called from within the class or any other class that derives it, but it may not be called from outside the class's "family". This is more for safety and organization than anything else, and you may prefer to omit the PROTECTED attribute.

## MyFile.clw

The method code in MyClass.clw starts very simply:

```
  MEMBER
  INCLUDE('MyFile.inc'),ONCE
```

Next come the Fetch and TryFetch methods, which seem deceptively simple:

```
MyFileManager.Fetch PROCEDURE(KEY K, ? F1 ,<? F2>,<? F3>, |
                                    <? F4>,<? F5>,<? F6>, |
                                    <? F7>,<? F8>,<? F9>)
  CODE
  IF SELF.AssignKeyFields(K,F1,F2,F3,F4,F5,F6,F7,F8,F9) <> K{PROP:Components}
    SELF.SetKey(K)
    MESSAGE('Too few components for MyFileManager.Fetch|' |
            & SELF.GetName(), 'Programmer Error!', ICON:Hand)
    HALT(0)
  END!IF
  RETURN PARENT.Fetch(K)

MyFileManager.TryFetch PROCEDURE(KEY K, ? F1 ,<? F2>,<? F3>, |
                                       <? F4>,<? F5>,<? F6>, |
                                       <? F7>,<? F8>,<? F9>)
  CODE
  IF SELF.AssignKeyFields(K,F1,F2,F3,F4,F5,F6,F7,F8,F9) <> K{PROP:Components}
    SELF.SetKey(K)  !Prepare internal queue to get Description
    MESSAGE('Too few components for MyFileManager.TryFetch|' |
            & SELF.GetName(), 'Programmer Error!', ICON:Hand)
    HALT(0)
  END!IF
  RETURN PARENT.TryFetch(K)
```

The only thing to note here is that the code uses the PROP:Components property syntax to determine the number of components in the key. Recall that GET requires that all key components be assigned. Doing it the "old way", this error would not be caught until someone (probably the user) observed strange behavior, followed by a potentially intense debugging session to determine the problem. Now the error will appear at runtime the first time that the code executes, catching it long before it gets to the user, and saving debugging time too. (Unfortunately, there is no way to catch this kind of error at compile time.)

The bulk of the logic is in encapsulated in the AssignKeyFields method:

```
MyFileManager.AssignKeyFields PROCEDURE(KEY K,<? F1>,<? F2>,<? F3>, |
                                              <? F4>,<? F5>,<? F6>, |
                                              <? F7>,<? F8>,<? F9>)
C BYTE,AUTO
N BYTE(0)
X BYTE,AUTO
F ANY               ! Field reference for value assignment
  CODE
  C = K{PROP:Components}
  LOOP X = 9 TO 1 BY -1
    IF OMITTED(2+X)  ! 1 for the class, and 1 for the key
      IF N           ! If count already remembered
        N = 0        ! Middle omits forbidden!
      END!IF
    ELSE
      IF N = 0       ! Haven't found last yet?
        N = X        ! This is the last
      END!IF
      IF X =&lt; C      ! Within key's components?
        F &= WHAT(SELF.Buffer, K{PROP:Field,X})
        EXECUTE X
          F=F1; F=F2; F=F3; F=F4; F=F5; F=F6; F=F7; F=F8; F=F9
        END!EXECUTE
      END!IF
    END!IF
  END!LOOP
  RETURN N
```

This code just counts backwards through the parameters, finding the first one that is not omitted. It remembers this field, and then continues to check for "holes" in the middle. It also assigns the values to the key component fields. Finally, it returns the count of assigned key components back to the caller.

Implementing this in an application could not be easier! Just go into an application's Global options, select the "Classes" tab, press the File Management button, and then change the File Manager to `MyFileManager`. Now the new methods are available, and everything else will work just like it did before!

As you can see, the ABC classes permit not only a more robust method of coding, but are also easily extensible. In future articles, I'll provide additional examples, in many cases adding templates to make you even more productive!

Download the source

---

*Mike Hanson is President of BoxSoft Development Inc, which produces the "Super" series of templates, distributed through Mitten Software. He has been creating add-on products for Clarion since his Public Domain Models for CPD 2.0 back in 1988. He's also written articles for every Clarion-related publication, and has spoken at numerous conferences and training seminars. If you have any questions, you can reach him via www.boxsoft.net.*

## Reader Comments

Add a comment

**Clarion Magazine**

# ClarionMag Subscription Price Increase Dec 1, 2004

Published 2004-10-25

Changes in foreign exchange rates over the last two years are making it necessary for Clarion Magazine to raise its subscription rates. We charge in US dollars for the convenience of our customers, but we are based in Canada, and over the last two years the Canadian dollar has risen substantially against the greenback, reaching an 11 year high in October of this year. As a result, our subscription rates will change on Dec 1, 2004, as follows:

| Product | Old Rate | New Rate |
|---|---|---|
| New subscription (includes all back issues, beginning with Feb 1999). | $125 | $150 |
| Current subscription renewal | $79 | $99 |
| Expired subscription renewal within three months (if your subscription expired more than three months ago, you will have to pay the new subscription rate again). | $99 | $125 |

All rates are in US Dollars.

To renew your subscription, please go to the Clarion Magazine Store.

## Reader Comments

Add a comment

**Will you consider subscription in Canadian currency?**