Clarion Magazine

Clarion News

Hurricane Katrina Relief

Clarion developers are among those affected by Katrina's devastation; at least one developer we know of has a flooded home and business losses, and of course many people are in desperate need. Donate to hurricane relief through the Red Cross.

- » SoftVelocity enters the blogosphere
- » C6 9048
- » TIFF-Change 3.23
- » Clarion Developer's Challenge Week #6 Winner Is Ian Cook
- » IP Driver Web Hosting
- » PrintWindow 1.00 Beta Build 94
- » FullRecord 1.02
- » NeatMessage 1.02
- » MS Visual Studio Launch and Free Product
- » CIA Factbook Download
- » xToolTipPro 1.0
- » Clarion Developer's Challenge Week #5 Winner
- » DevDawn Blog
- » FinalStep 2.0 Split Up - NeatMessage Emerges
- » TipLink Tool Suite
- » UK Clarion User Group Meeting Nov 21
- » Whitemarsh Metabase 6.8
- » Three Way Tie In Week Four Of Clarion Developer's Challenge
- » xCheckTPS 1.04

- » SysTree C6.2 Compatible
- » BSPrintList 1.0
- » dpQuery 2.06
- » Up & Up 1.2
- » iQ-XML 1.10
- » solidsoftware Atom Feed
- » solidsoftware RSS Feed
- » SysList Clarion 6.2 Compatible
- » xTipHotKey 2.9
- » Oz DevCon Survey
- » Data Down Under Distributes SetupBuilder
- » Clarion Developer's Challenge Week #3 Winners
- » Free CWPlus Wallpapers

[More news]

Podcast

[Track lists, more podcasts]

Latest Free Content

Clarion and the War on Terror

Ozzie Paez discusses Clarion's use in support of Homeland Security research conducted over the past two years, and Clarion's demonstrated advantages supporting evolving research. When it comes to delivering quality applications, productively and competitively, the Clarion paradigm offers advantages that have proven effective in supporting the national efforts against terrorism.

[More free articles]

Latest Subscriber Content

Mixing Clarion With .NET, Part 5

Wade Hatler's series on using Clarion and .NET together continues with a discussion of the tricky business of passing groups.

Posted Tuesday, October 11, 2005

PDF for September 2005

All Clarion Magazine articles for September 2005 in PDF format.

Posted Tuesday, October 11, 2005

Handling Windows With Hundreds Of Controls

If you've ever had to deal with a window with hundreds of controls, you know how difficult a task this can be. Maarten Veenstra demonstrates a variety of techniques for placing and creating controls, as well as some data validation tips and tricks.

Posted Wednesday, October 12, 2005

Internationalization Tools Standards: Learning from an ABC Calendar Workaround

Anyone who has internationalized a large application knows that one of the more time consuming tasks is dealing with third party products which attack internationalization in ways that do not allow languages choices at run time. Even in a pure ABC context, creating multi-language applications is challenging but generally possible, although there is a notable exception in the ABC calendar classes. Phil Will shows how to prepare your code for translation, using the misbehaving calendar classes as an example.

Posted Thursday, October 20, 2005

Clarion and the War on Terror (free article)

Ozzie Paez discusses Clarion's use in support of Homeland Security research conducted over the past two years, and Clarion's demonstrated advantages supporting evolving research. When it comes to delivering quality applications, productively and competitively, the Clarion paradigm offers advantages that have proven effective in supporting the national efforts against terrorism.

Posted Thursday, October 20, 2005

A Tree in a Page Loaded Browse: the Sequel, Part 1

Browse boxes formatted as trees are handy for all sorts of things. File loaded trees are easy; page loaded trees, however, can be a nightmare. In this series David Podger and Deon Canyon build on an earlier approach demonstrated by Ronald van Raaphorst. Part 1 of 3.

Posted Thursday, October 27, 2005

Exporting APPs and DCTs to XML

XML is everywhere these days; everywhere, that is, but in your DCTs and APPs. But as Harley Jones shows, exporting your APP or DCT to XML can be a great benefit, for documentation, and for other purposes. Part 1 of 2.

Posted Friday, October 28, 2005

[Last 10 articles] [Last 25 articles] [All content]

Printed Books & E-Books

E-Books

E-books are another great way to get the information you want from Clarion Magazine. Your time is valuable; with our e-books, you spend less time hunting down the information you need. We're constantly collecting the best Clarion Magazine articles by top developers into themed PDFs, so you'll always have a ready reference for your favorite Clarion development topics.

Printed Books

As handy as the Clarion Magazine web site is, sometimes you just want to read articles in print. We've collected some of the best ClarionMag articles into the following print books:

- Clarion 6 Tips & Techniques Volume 1 - ISBN: 0-9689553-8-X
- Clarion 5.x Tips and Techniques, Volume 1 - ISBN: 0-9689553-5-5
- Clarion 5.x Tips and Techniques, Volume 2 - ISBN: 0-9689553-6-3
- Clarion Databases & SQL - ISBN: 0-9689553-3-9

We also publish Russ Eggen's widely-acclaimed Programming Objects in Clarion, an introduction to OOP and ABC.

From The Publisher

About Clarion Magazine

Clarion Magazine is your premier source for news about, and in-depth articles on Clarion software development. We publish articles by many of the leading developers in the Clarion community, covering subjects from everyday programming tasks to specialized techniques you won't learn anywhere else. Whether you're just getting started with Clarion, or are a seasoned veteran, Clarion Magazine has the information *you* need.

Subscriptions

While we do publish some free content, most Clarion Magazine articles are for subscribers only. Your subscription not only gets you premium content in the form of new articles, it also includes all the back issues. Our search engine lets you do simple or complex searches on both articles and news items. Subscribers can also post questions and comments directly to articles.

Satisfaction Guaranteed

For just pennies per day you can have this wealth of Clarion development information at your fingertips. Your Clarion magazine subscription will more than pay for itself - you have my personal guarantee.

Dave Harms

Clarion Magazine

Clarion and the War on Terror

by Ozzie Paez

Published 2005-10-20

Developing applications for a stable environment and developing applications to support research can be as different as developing with Clarion and non-RAD tools. Traditional application development allows the programmer to define most key requirements up front, validate and adjust them along the way, test and deploy within a generally known set of conditions. Developing in support of research, however, often involves at least as many unknowns as knowns, since the research itself is bound to affect initial assumptions and considerations.

This article discusses Clarion's use in support of Homeland Security research conducted over the past two years, and Clarion's demonstrated advantages supporting evolving research. This experience represents a specific case within the evolution of a programming paradigm, which I presented at a DevCon 2004 session titled *Evolving Today's Development Paradigms to Assure Software Lifecycle Quality and Security in Cyberspace*. That presentation focused on how Clarion's approach can be adapted to address many of the challenges facing programmers in developing quality products, within a highly dynamic environment. I will summarize this topic in the second part of this article; first, some background on how Clarion's RAD capabilities have been useful in the War on Terror.

Background

In the spring of 2003 I was asked by a colleague to support a number of Homeland Security research projects being conducted through the Network Information and Space Security Center (NISSC) of the University of Colorado at Colorado Springs (UCCS). The research included evaluations of existing systems that could be adapted to various aspects

of the War on Terror, as well as some initial modeling of the type of information required to conduct security threat assessments.

One of my deliverables during that time was a preliminary model that would illustrate a conceptual information management system to support these efforts. The research conducted during this phase included potential targets, mode of attacks, logistical/costs requirements and a variety of other factors, which were expected to play a part in the enemy's decision making process.

The initial NISSC project evolved into additional studies focusing on integrating security risk assessments into Environmental, Health and Safety programs. The results of these efforts, which included information modeling, was a Security, Environmental, Health and Safety assessment model/process, which was piloted through additional grants. The results of these studies were later published in Environmental Quality Management (Spring 2004) and Disaster Prevention and Management (Volume 14/No 1/2005, UK).

Supporting research into what was then an evolving practice meant that data and information models were essentially in a constant state of flux. Their evolution was driven by a combination of changing use cases and improved understanding of the factors affecting the enemy's thinking. The following sections describe the efforts involved in more detail, and the role which Clarion's highly integrated RAD environment played in making it possible to quickly respond to the highly dynamic nature of the underlying data and user interface requirements. Clarion contributed greatly to the delivery of targeted solutions, from utilities to conceptual designs, that supported our work with minimal disruptions from undocumented features, i.e. bugs.

Terrorism and Databases

Terrorism did not begin with 9/11; instead, that awful day brought much greater awareness of the threats we faced and the real urgency of our response. From an information management perspective, a need immediately arose to capture as much information as possible about the terrorists and their tactics, methods and strategy.

One of the most challenging aspects of the war on terror has been to identify the threats posed by the enemy, without putting at risk the civil rights of individuals based on group factors, i.e. race, nationality, religion, etc. Researching threat profiles from radical Islamist organizations meant going back in history to identify the common characteristics of the various movements, which ultimately evolved into what is loosely referred to as AlQaida. Specifically, the research focused on these organizations' membership, their behavioral

characteristics, demonstrated capabilities, choices of weapons, means of attacks, strategy, tactics and other factors, which if properly analyzed could assist national authorities in identifying and neutralizing threats before they are carried out.

These efforts attempted to understand how the enemy made decisions, and how those decisions culminate in a resultant event, such as the attacks against the USS Cole, the Embassy bombings in Africa, two attacks against the World Trade Center and similar events across the globe, as we recently witnessed in London.

From an information management perspective, databases had to be created to contain the data uncovered during research, the application of models and the results of pilot programs. Data sources included published reports, Internet sources, and interviews with various government representatives and original research.

Diverse Sources and Dynamic Requirements

Designing a support system to facilitate research based on a diverse, and more importantly changing, set of parameters, all within a highly compressed timeframe, would stress any development team. The individuals with whom I continue to collaborate, and those whose work we have studied, are equally diverse in their areas of expertise and computer skills. These people include Environmental, Health and Safety practitioners, process analysts, experts in organizational leadership, historians, psych majors, statisticians and emergency response managers. Their professional backgrounds included academia, US military, corporate management, consulting and emergency response.

Such variety in the researchers and in the nature of the subjects being studied translates into constant change and adaptation of databases and related user interfaces.

Development Strategy

The nature of the dynamic environment and evolving requirements of these research projects precluded the development of full requirements definition in the traditional sense. There were initial efforts to create core data models and these proved useful, but they were just a starting point.

An analysis of the efforts to be supported and the dynamic nature of the data made it clear that Clarion’s Data Dictionary would have to serve as the nerve center of the effort. What made this possible, of course, is Clarion’s ability to generate code that implements dictionary requirements with minimal programmer involvement, provided that the

dictionary is fully leveraged through detailed definitions of fields, default values, relationships and other parameters.

An interesting strategy for dealing with SQL backends under this development and operating environment was to minimize the work done at the database level, before the underlying architectural relationships were validated. Thus, keys and parent-child relationships were often defined and validated through the dictionary/application, before implementation was shifted to the database itself.

Where research tools were intended for use over a short duration, such as to demonstrate specific use-cases and underlying models, database relationships and constraints were kept within the Clarion application, saving valuable time. We didn't have the time to define, develop and test backend components, i.e. triggers, stored procedures and views. Implementing, modifying and testing these components would have required us to take the database off line repeatedly, risk introducing coding errors, and significantly delay most deliverables.

By relying on the dictionary and the application, the desired behaviors (relationships, constraints, cascaded actions) could be easily implemented and the application updated and tested as requirements changed, without significantly affecting the underlying database and its availability.

Quality and Productivity

In almost all software engineering/programming situations, it is the programmer that introduces a majority of the errors. In the past, error rates were related to lines of authored code. Thus, for projects where the error rate was estimated as one per five hundred lines of authored code, a program with one hundred thousand lines of code should be expected to contain about two hundred errors.

Most errors are easily identified and corrected due to the manner in which they affect program behavior. Others, however, may not be identified unless a specific set of conditions, which might not be included in the test plan, exist. These are often the source of unexpected, nasty surprises, which almost all programmers have and will repeatedly experience during their careers.

Part of the strategy for managing dynamic change in support of the research described above was to limit the amount of embedded or hand-written code. In this environment, quality and productivity was provided by the general data architecture, the use of the dictionary and the template-generated code. The dictionary became the heart of the

research support system precisely because it reflected the underlying data model(s), which in turn reflected the environment, organizations and people being studied. The dictionary actually became an important research tool in itself, by revealing relationships and conflicts between various aspects of the data and information being captured. As a result, it served as a tool that helped frame the information/data being gathered in a coherent and consistent manner.

In the end, the strategy described above resulted in very high productivity and quality, measured as the ability to quickly adapt the underlying data architectures and screen designs based on new information, requirements, distinctions and user needs. Software bugs were relatively few and did not significantly impact system availability and usability. Since the screens themselves were often all or part of the deliverables, use-case methodologies were defined and implemented through Clarion's Designer, in order to contextually present the underlying data/structures in different ways for different audiences. Thus, not only were systems and tools quickly created, Clarion's productivity allowed us to demonstrate different user interface designs in to support varying audiences. For example, one person might be studying high-end threats such as Weapons of Mass Destruction, while others may need to drill down within a particular type of WMD to the logistics, knowledge, skills and other factors associated with its construction, deployment and use.

Lessons Learned for Quality and Security

Clarion for Windows proved capable of supporting research within a highly dynamic and difficult to predict information management system design environment. Its ability to quickly translate data models into specific table designs and overlay relationships, constraints and behaviors based on dictionary directives proved ideal for this purpose. Since almost all development was against SQL based databases (MSSQL and Oracle), using the dictionary to prototype and validate the architecture, relationships and constraints saved countless hours.

In parallel, Designer's ability to translate dictionary directives into user interface designs and behaviors, combined with its advance code generation capabilities proved itself in the implementation of client server and web-based systems and tools. At a higher level, templates were leveraged as much as possible to minimize hand coding. Where appropriate, new templates were obtained, designed or existing templates modified to implement recurring behaviors.

The lessons learned from the research projects described above support the concept of

evolving traditional programming paradigms to better cope with today's many technical and business challenges. These challenges include the ability to deliver better solutions, with higher quality, greater flexibility and improved cost-benefit levels.

As I write this article, it is being reported that Microsoft expects to outsource over ten thousand programming jobs to China over the next decade. While no one I know is seeking to keep the benefits of technology away from other societies, neither should we remain complacent as a growing number of companies work to commoditize and export a growing share of the most creative and critically important technology jobs in our economy. Exporting jobs is actually a two edge sword because, while on the one hand a company can lower per-employee unit costs, on the other communication between the worker and client (internal or external) almost always suffers, particularly over long distances, many time-zones and a wide cultural divide. Poor communication remains a critical root cause behind poor program design through poorly defined, understood and implement user needs; thus, real incentives exist for development to remain as close to the customer as possible.

The Clarion paradigm can successfully integrate evolving development paradigms with effective communications and complimentary programming team skill sets to offset the blind drive towards cheaper labor being promoted within some business circles. In the end, most clients do not want to know how much employees get paid per hour. What they want are solutions that address their needs, within a competitive cost-structure and with the quality necessary to ensure reliability and security.

For many areas in Government, security and quality are even more important, particularly, where applications are accessed through Intranet, and increasingly Internet, portals. Microsoft has been miraculous in its ability to deliver products to market, but that does not mean that it has a corner on ideas and concepts. And, when it comes to delivering quality applications, productively and competitively, the Clarion paradigm offers advantages that have proven effective in supporting the national efforts against terrorism. Clarion can prove even more effective in growing future opportunities for the next generation of American programmers and solution providers.

Ozzie Paez is a systems engineer with over 25 years experience as a programmer and database architect using a variety of languages including Fortran, Modula 2, Pascal, Basic, Assembly and Clarion. He has been using Clarion since 1987 and became a Certified Clarion Developer in 1999. Mr. Paez's work history includes the US Military, Civil Nuclear industry, Department of Energy's Weapons Complex, Telecommunications,

Technical Support and Security Center Management. He currently works as a systems engineer, solutions provider, software quality assurance lead and security researcher for a variety of clients, including the US Air Force. He has published a variety of articles in the US and Europe on issues ranging from business management, corporate leadership, regulatory compliance and quality, to his current focus developing models for conducting threat assessments in the war on terrorism.

Reader Comments

Add a comment

Clarion Magazine

Mixing Clarion With .NET, Part 5

by Wade Hatler

Published 2005-10-11

So far in this series I've covered calling .NET from Clarion, using .NET controls in a Clarion window, and the basics of calling Clarion from .NET. Simple Clarion procedures are easy to call from a .NET application, but things get trickier when you start passing and returning values. In this article I'll cover the ins and outs of passing groups.

The way Clarion handles `GROUP` parameters and the way `.NET` handles the closest analogs (`struct` or `class`) are different. Clarion always passes `GROUP` parameters by reference, so in your prototype `GROUP`, `*GROUP`, `SpecificGroup` and `*SpecificGroup` all act identically. The more specific type of group parameter gives the compiler a hint to produce an error if a different group type is passed, but doesn't change the semantics of how the actual call is made. Clarion groups are also packed, which means the fields appear in the order that they appear in the source code with no gaps.

.NET on the other hand handles `struct` and `class` differently. By default, the runtime can order the actual fields in the group in any convenient order, and all fields will appear on a word boundary, meaning there could be gaps between the fields, which may or may not be in the order you declared them. .NET also passes a `struct` by value by default, and a `class` by reference by default.

Two other factors complicate this. First, for interop purposes `pinvoke` makes a copy of the fields for any `struct` or `class` that is passed to an unmanaged DLL, and then if the field was passed using reference semantics it copies the changed values back after the return. Second, Clarion passes all `group` parameters using three actual parameters, the address of the group itself, plus the length of the group, and the offset of a descriptor that tells how the group is laid out. This means that to pass a group to Clarion, we have to either get rid of the `GROUP` parameter type, or figure out how to pass those extra two fields. Passing the length wouldn't be a huge problem, but finding the address of the descriptor block in .NET would be difficult or impossible to do reliably. Passing a bogus descriptor block might work, but then again it might not, or worse yet it might work until the next release of the compiler. In the end, I decided the best path was to eliminate the `GROUP` semantics because that can be done easily and reliably.

For an example, I'll use this simple group that's defined in `ShareDLL04.inc`:

```
TestGroup                   group, type
ByteVal                        byte
ShortVal                       short
LongVal                        long
RealVal                        real
                            end
```

Lay out the struct

The first step is to lay out a group in .NET to mirror the group in Clarion. This is fairly straightforward, and more or less the same thing as you've probably done in reverse for years to use the Windows API. The only key is that before the `struct`, you have to add an attribute to tell it to lay the structure out to match Clarion's layout. All you have to do is paste in an attribute above the `struct` that never changes. The attribute is:

```
[StructLayout(LayoutKind.Sequential, Pack=1, CharSet=CharSet.Ansi)]
```

The complete struct to match the group above is:

```
[StructLayout(LayoutKind.Sequential, Pack=1, CharSet=CharSet.Ansi)]
public struct TestGroup
{
   public CWByte  ByteVal;
   public CWShort ShortVal;
   public CWLong  LongVal;
   public CWReal  RealVal;
}
```

The attribute above makes instructions about how the `struct` is to be marshaled using pinvoke. It doesn't tell the CLR how to actually lay out the struct in memory. Note that you also have to mark both the `struct` and every field as public. In C#, everything not explicitly marked as public is private.

Nix the GROUP

The next step is to get rid of the `GROUP` semantics of the call without losing the ability to handle the `GROUP`. The key is that you have to change the parameter to a `LONG`, and then create a reference variable as a pseudo-parameter on startup to convert the long back to a group reference. This allows the compiler to piece together all the information that's usually passed in the extra parameters without requiring .NET to generate the parameters. There are two subtly different flavors of this you can use, depending on whether you want your Clarion procedure to be able to change the values in the passed `struct` in the .NET side. In Clarion, there are no pass-by-value semantics for groups, but in .NET you can do it. If you use the pass-by-value semantics, the code will be slightly faster because the marshaler won't have to update the .NET structure to match any changes you may or may not have made to the group.

The first thing to do is get rid of the `GROUP`. You do this by simply changing the `GROUP` or named group parameter to a `LONG`, and anywhere you call that procedure in your codebase, you need to pass the `address` of the group. That means you would change these:

```
TakesAndReturnsGroup       procedure(TestGroup pGroup), pascal
...
LocalGroup            like(TestGroup)
...
TakesAndReturnsGroup(LocalGroup)
```

to

```
TakesAndReturnsGroup       procedure(long pGroupAddress), pascal
...
LocalGroup            like(TestGroup)
...
```

```
TakesAndReturnsGroup(address(LocalGroup))
```

Once you've changed the prototype, you have to add a bit of code to the procedure to decode the `LONG` value. In the case where you want to exactly match the Clarion semantics (meaning pass by reference and changes made in Clarion will show up in .NET), you change this:

```
TakesAndReturnsGroup procedure(TestGroup pGroup)
```

to

```
TakesAndReturnsGroup       procedure(long pGroupAddress)
pGroup                     &TestGroup
   code
   pGroup &= (pGroupAddress)
```

Once you've done this, the group will *appear* to work just like it does in Clarion. In Clarion, it is actually doing the same thing, but in an indirect way. Instead of passing the definition on the stack, Clarion is filling in the missing pieces when you do the reference assignment. Note the parenthesis around the `pGroupAddress`. This is an undocumented Clarion trick that assigns to a reference variable using nothing but an address. It doesn't do any compile time checking, but if the address points to a valid instance of what the reference variable is expecting it will all work out OK.

The .NET Side

On the .NET side of things, to make this work you have to use call-by-reference semantics. Every .NET language implements this a little bit differently. For C#, the key is to add the `ref` keyword before the parameter type in the declaration, and then you have to also add the `ref` keyword when you call it. This is a feature of the C# language to try to make it obvious when you're calling the code that something in your parameter might be changed. I think is generally a good idea. I'll trade a little typing for subtle bugs any day.

You've already seen the `struct` layout above. Here's the pinvoke prototype:

```
[DllImport("SharedDLL04", EntryPoint="TAKESANDRETURNSGROUP")]
public static extern void TakesAndReturnsGroup(ref TestGroup pTestGroup);
```

To call it, you also have to use the `ref` keyword like this:

```
CWLink.TestGroup localGroup = new CWLink.TestGroup();
localGroup.ByteVal  = 12;
localGroup.ShortVal = 123;
localGroup.LongVal  = 1234;
localGroup.RealVal  = 12.34;

CWLink.TakesAndReturnsGroup(ref localGroup);
MessageBox.Show(String.Format("Returned Doubled From Clarion\n{0}\n{1}\n{2}\n{3}."
                            , localGroup.ByteVal, localGroup.ShortVal
                            , localGroup.LongVal, localGroup.RealVal));
```

You'll find a button in the source `Interop04` that does exactly this.

Wrapper Procedures

If you're making a brand new procedure, or one that isn't used very often, you can do this exactly as described, particularly if the procedure will never be called from Clarion. If you want to use an existing procedure without having to change your Clarion code, just make a wrapper function to expose to .NET. For example, you could solve this same problem by making this wrapper to call from .NET.

```
TakesAndReturnsGroup_Wrapper procedure(long pGroupAddress)
pGroup                       &TestGroup
   code
   pGroup &= (pGroupAddress)
   return TakesAndReturnsGroup(pGroup)
```

One-Way Groups

Everything shown above will work and for the most part you'll want to stick with it. If you get into a situation where you really need performance, you can make the group transfer one-way only. This will have the effect of changing the pass-by-reference semantics to pass-by-value. To do that, simply remove the `ref` keyword on the .NET side, and expand the assignment of the pGroup by one level of indirection, by using the `address` function like this:

```
pGroup &= address(pGroupAddress)
```

This will work just like before, except the changed values won't be copied back into the `struct` in the .NET side.

Passing Groups to .NET

There are a couple of ways to pass `groups` to .NET that I'll cover in a later segment. You can't do it with the OLE control, but you can have the OLE control call a .NET method that immediately calls back to Clarion with the group to be filled in. You create a procedure in Clarion that takes a group, and you expose that procedure to .NET. Then make a method in .NET that calls this procedure to fill in the values. Then call that .NET method and viola... you're there. It's exactly the same paradigm that Windows callbacks use, so it isn't even all that much of a hack.

Do you really need the group

One other thing to ask yourself is if you really need the whole group. Lots of times you'll pass a group in Clarion because it's already defined, and it's handy. If all you need is a few fields, you can quite easily just create a wrapper procedure to pass or return just the fields you need. This will even impress the private data Nazis, so keep it in mind.

Wrapup

This is the last of the basic data types that need to be passed. In the next segment, I'll cover a couple of ways to get rid of the `OLE` control to get better performance and more flexibility, and then I'll move on to advanced window and control tweaking

Download the source

Wade Hatler has been around Clarion so long he got the very first patch release for Clarion 1.0 for DOS. That was back when Clarion had a dongle, no compiler and no templates. Wade co-owned IntelliScan, a company producing third-party Clarion products for several years. For the last 10 years he has worked for IMPAC Medical Systems creating software for the management of Cancer Therapy. He recently completed a 3-year, 12,000 mile, 12 country bicycle tour of the world, during which he carried a laptop and worked about half-time. He lives with his wife and daughter near Seattle. You can reach Wade at *WadeHatler@wademan.com*.

Reader Comments

Add a comment

Clarion Magazine

Handling Windows With Hundreds Of Controls

by Maarten Veenstra

Published 2005-10-12

I have an application that calculates the signal strengths in a cable TV network. These signals are boosted and spread out over the country by cascading amplifiers. The output of each amplifier is, via a set of splitters, directed to one to ten other amplifiers. In order to calculate the signals, the app needs to know which components and cables are used to interconnect these amplifiers. To make this easy for the end user, this information can be entered on one form for ten amplifiers simultaneously.

The form I designed for this task (see Figure 1) contains close to 600 controls. On top of that, the components must be picked from a FileDropBox (DropDownListBox), and the cables must be looked up from a browse that is called from a button. It already needs a small miracle to create such a window in the IDE, let alone adding 120 FileDropBoxes *that all access the same file* and 50 lookup buttons *that all call the same browse.* Can you imagine 119 alias files and generated source with 120 views and 120 queues?

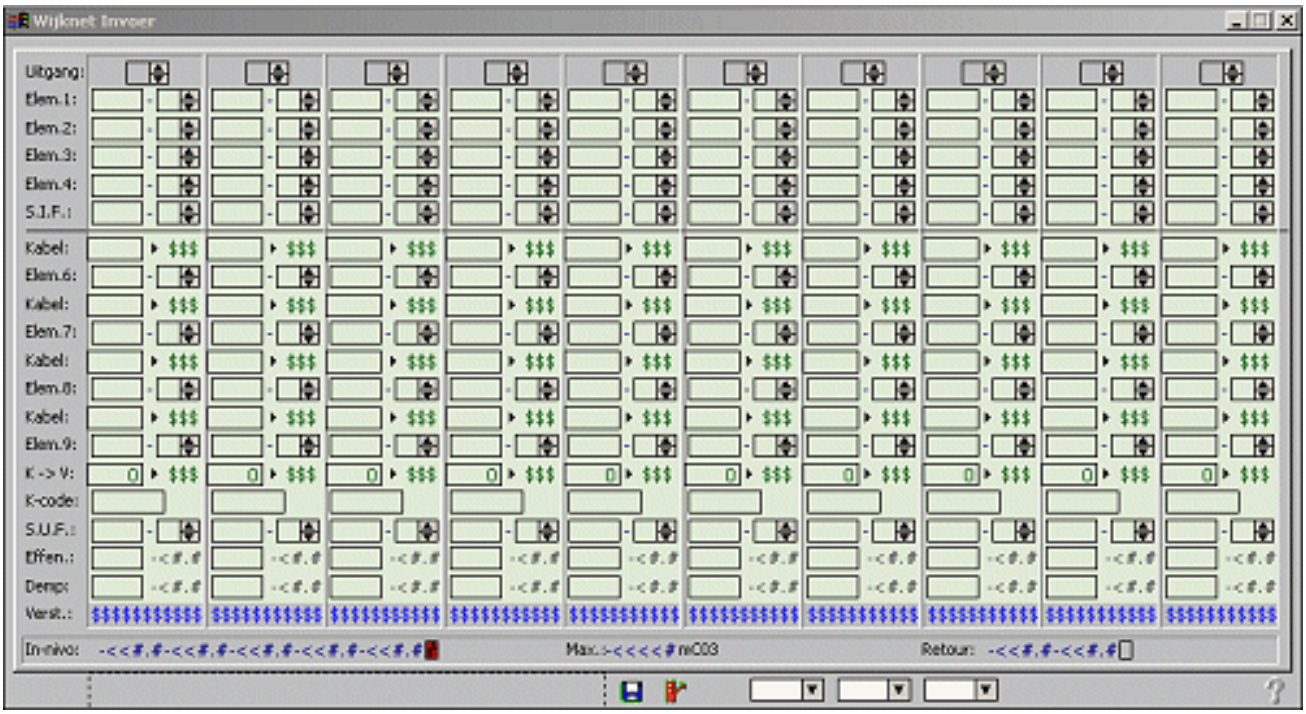

Figure 1. A form with almost 600 controls (view full sized image)

I used several different techniques to get all those controls on one window with a minimum of effort, and without breaking the IDE. These include:

- Creating one set of controls in the formatter, and copying/pasting in source
- Creating local procedures for often-used code
- Using a queue of field equates to simplify field editing
- Using one FileDropBox for all lookups
- Creating some controls at runtime

The result was a window that the IDE and the compiler could both handle, and which wasn't all that difficult to create.

Creating the window

Since the form updates ten records from the `Amplifier` file, I temporarily store those in local variables, which are dimensioned ten times (for each component per amplifier) by ten times (for ten amplifiers). These variables have a prefix of `SGA:` (ScreenGroupArray) and are used to populate the form. Since I'm dealing with ten records, I can make a symmetric form, with ten identical columns. I start out by populating a `GROUP` and inside that group all controls needed for the first amplifier. Then I close the window formatter and go into its source via the […] button. There I identify this `GROUP` and make a copy below the original. Since all the controls are dimensioned variables, I change the index from 1 to 2. Then I open

the window formatter again and select the second GROUP, by tabbing or F12 (because it now exactly overlays the first group), and drag it to the right. I repeat this process for all ten GROUPs (amplifiers). The result looks like Figure 1. The light green background comes from a filled BOX at the top of the screen structure.

Local procedures

In order to keep my code clean and maintainable, I create local procedures by deriving the WindowManager class. Press the Window Behavior button on the window properties and go to the Classes tab. There you tick the Derive checkbox and press the New Class Methods button. You can now insert a name and prototype for your own method (local procedure). This new name will then appear (pink) in the embed-tree. You call these methods with the syntax SELF.*MyMethodName(Parm)*.

A queue of field equates

Dimensioned variables are easy to lay out repetitively using the technique I described above, and it's easy to process the edits as well, provided you know the field equate (FEQ) that corresponds with each variable. I built a queue with all my FEQs, declared as:

```
FeqQ    QUEUE,PRE(FQ)
Feq       LONG     !Control FEQ
Col       SHORT    !Vertical Column no
Row       SHORT    !Horizontal row no
Type      SHORT    !Control type
          END
```

This queue is filled the conventional way in my derived method, SetupWindow(), which is called from ThisWindow.Init, after opening the window. Here's the relevant code from SetupWindow():

```
LOOP FQ:Col = 1 TO 10
  FQ:Feq  = ?SGA:ElmCode_1_1 + ((FQ:Col - 1) * 53)
  FQ:Row  = 1
  FQ:Type = 1
  ADD(FeqQ, +FQ:Feq)
  ! Same for all 32 'focusable' controls in one column
END
```

In the above code `?SGA:ElmCode_1_1` is the field equate corresponding to the first array element.

The floating FileDropBox

The components that the user needs to enter can be a Splitter, an Attenuator or an Equaliser. These are stored in the `Elements` table, and must be selected via the FileDropBox.

A FileDropBox has a `USE()` variable and a Target variable in which it places the `SysID` of the picked choice. So I created two local variables: `LST:ElmCode LIKE(ELM:ElmCode)` and `LST:ELMid LIKE(ELM:ELMid)`. This way I could copy the selected value out of the temporary variable and into the array element, as needed. I populated a FileDropBox for the `Elements` table on the window, using these two local variables, between the Cancel and Help buttons, and set the `HIDE` attribute. I also declared a local `LONG`, `LST:Feq`, to hold the FEQ of the currently selected control.

Entering and validating data

Since I had so many controls, and so many similar controls, it didn't make sense to have embed code for each control. Instead I use my queue of field equates (FeqQ) to determine what actions need to be taken for any given control. In `ThisWindow.TakeSelected` I do the following:

1. Check if the selected `FIELD()` is in the FeqQ
2. Use the `FQ:Type` to determine what sort of control is selected.
3. In the case of an Element-entry
    a. Copy the `ElmCode` and `ELMid` from the `ScreenGroupArray` to the FileDropBox's variables
    b. Lookup the `ELMid` in the FileDropBox's queue, to position the choice of the FileDropBox
    c. Set the FileDropBox's X and Y coordinates to the same coordinates of the selected entry
    d. Unhide the FileDropBox and select it.

The end user sees the entry-field changing into a FileDropBox at the moment (s)he selects it, displaying the same information as the entry-field did before.

In the `EVENT:Accepted` of the FileDropBox (`?LST:ElmCode`)

1. Use the `LST:Feq` to retrieve the correct record from the FeqQ
2. Copy the `ElmCode` and `ELMid` from the FileDropBox back to the `ScreenGroupArray`
3. Hide the FileDropBox again
4. Select the next control, after `LST:Feq`

Perfect, my Floating FileDropBox follows the user's selections so the user can select an Element at the place where it is supposed to be entered. But what if the user doesn't select an element and simply tabs off? Then the next control in the screen-structure's sequence is selected, which happens to be my Help button!

Getting the FileDropBox right

I have been playing with `PROP:Follows`, setting the next control (`LST:Feq + 1`) to follow the `LST:ElmCode (FileDropBox)`, but that only worked for one control. It seems that when you set this property on the FileDropBox once, it won't let go and thus has the effect that you could not move the FileDropBox to another control anymore; it becomes stuck.

I thought that if I checked if the Help button became selected (in `ThisWindow.TakeSelected`) *and* `?LST:ElmCode{Prop:Hide} = False` I could safely assume that the user tabbed of the FileDropBox; I would then issue a `SELECT(LST:Feq + 1)`. Wow, that worked! I also had to allow for <Shift><Tab>, where the Cancel button will be selected, on which I performed the same trick. All's well, I thought. Until I clicked the Cancel button when the FileDropBox was visible: The window didn't cancel but the FileDropBox moved up one entry-field! And it moved down when I clicked the Help button. Now, I cannot go and explain the customer she has to click the Help button 120 times (until the FileDropBox is hidden) in order to call the help!

I need to know when the user tabs off my FileDropBox. For this reason I have asked SoftVelocity for an `EVENT:LoseFocus` on a control because you can tell where the focus is going to but no way where it came from. As a workaround I created two dummy-entries as "bounce"-fields, placed around the FileDropBox. The one in front I called `?BounceUp` and the one after I called `?BounceDn`. You can choose any control that accepts focus as a dummy field but some, e.g. a flat, transparent, button w/o text will still raise visibly when selected. So I chose a transparent entry control of 6x8. You can still see the cursor flash for a split-second when it is selected but that's hardly noticeable. I have placed them after the number strings in the statusbar so it looks like a screen-refresh when they flash. I also (un)hide them along with the FileDropBox.

Instead of checking the `?Help` and `?Cancel` buttons I now check for my bounce fields and use a "GoingUp" flag to indicate the direction. Since the FileDropBox already controls its own movement I simply post an `EVENT:Accepted` to the FileDropBox (`?LST:ElmCode`). And now my screen navigating works as it should.

Well, almost. When the user clicks a control that doesn't need the FileDropBox, the FileDropBox stays visible. This is solved by simply hiding the FileDropBox when the FileDropBox is *not* selected, in `TakeSelected`.

There's still one problem left: Once you make a selection from a FileDropBox control, you can never undo that selection, i.e. clear the value you have selected. On a "normal" form you can add a Clear button, but on this window that is impossible. So I alerted the `EscKey` on the FileDropBox control and in its `EVENT:Alert` embed I do the following:

- Check for the EscKey
- Hide the FileDropBox
- Use the `LST:Feq` to retrieve the correct record from the `FeqQ`
- Clear the correspondent index in the `ScreenGroupArray`
- Select the next entry field

Of course I could also have solved this problem by using a FileDropCombo instead of a FileDropBox but, in my code, the FileDropBox is controlling its own movements. Therefore I need `Event:Accepted` to be fired. This is handled by setting `PROP:Touched`. The FileDropCcombo has a problem with PROP:Touched, as I have reported in the Problem Tracking System (see #10391).

After this bug is fixed I will eventually change my FileDropBox controls in this window for FileDropCombos.

The attentive reader has seen three FileDropBox controls in the lower right of the window in Figure 1. The bottom two rows are only allowed to accept an Equaliser ("Effen.:") or an Attenuator ("Demp:") thus these FileDropBoxes have a filter on "Equaliser" and "Attenuator" and are handled exactly the same way as the unfiltered FileDropBox, described above.

The 50 lookup-buttons

The small solid right-arrows, on the "cable-rows", are buttons 7x10 with the `ICON:VCRplay`.

In `ThisWindow.TakeAccepted` (top of loop) I do the following:

1. Check if the `ACCEPTED()` field is in the FeqQ
2. Use the `FQ:Type` to determine what sort of control is accepted.
3. In the case of the Cable Type lookup button
    a. Set `GlobalRequest` to `SelectRecord`
    b. Call the lookup browse
    c. Check `GlobalResponse` if a selection has been made
        i. Copy the Cable-File fields to the `ScreenGroupArray`
        ii. Select the next entry

That's it! Lookup code for 50 buttons.

Figure 2 shows the window at runtime.

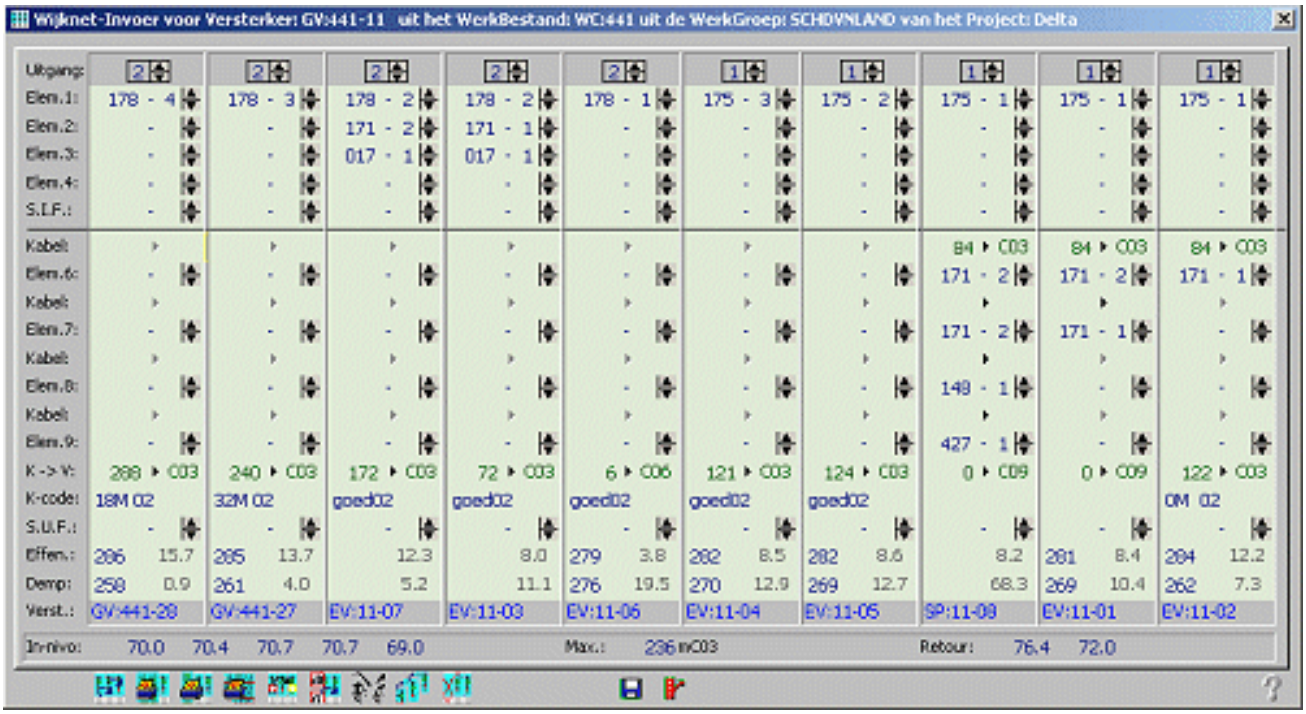

Figure 2. The window in Figure 1, at runtime (view full sized image)

You'll notice a row of buttons on the left-bottom of this window. They perform some specific task with the already entered data. I could not create those buttons in the window-formatter because that would crash the IDE, so I had to create them at runtime.

Dynamically created buttons

The `CREATE()` command can (optionally) take a parent as a parameter. Since I didn't want my created buttons to alter my tab-order or bounce fields of my symmetric screen, I added a

GROUP to the lower left bottom of this window, as you can see in Figure 1.

The problem with the CREATE() command is that the FEQs are known by Designer, and don't show up in the ACCEPT tree of the window, which makes it difficult to embed code.

What I did is assign the FEQs at design-time (in the DATA section):

```
?Fbutton1 EQUATE(601)
```

etc. up to

```
?Fbutton9 EQUATE(609)
```

Then, in my SetupScreen method (which is called from ThisWindow.Init), I create the buttons:

```
CREATE(?Fbutton1,CREATE:Button,?ButtonGroup)
?Fbutton1{PROP:Width}  = 16
?Fbutton1{PROP:Height} = 14
?Fbutton1{PROP:Xpos}   = 40
?Fbutton1{PROP:Ypos}   = 272
?Fbutton1{PROP:Icon}   = '~WNverd1.ico'
?Fbutton1{PROP:Tip}    = 'Resort the input (F4)'
?Fbutton1{PROP:Key}    = F4key
?Fbutton1{PROP:Flat}   = True
?Fbutton1{PROP:Hide}   = False
!Up till CREATE(?Fbutton9,CREATE:Button,?ButtonGroup)
```

Finally, in ThisWindow.TakeAccepted, priority 3030, I simply embed this code:

```
OF ?Fbutton1              !Resort (F4)
OF ?Fbutton2              !Print 1 Amplifier (<Shift>F5)
OF ?Fbutton3              !Print all Amplifiers (<Ctrl>F5)
OF ?Fbutton4              !Print This screen (<Alt>F5)
OF ?Fbutton5              !Copy Element (F7)
OF ?Fbutton6              !AutoPlace Eq./Atten. (<Shift>F8)
OF ?Fbutton7              !Call Drawing Viewer (F9)
OF ?Fbutton8              !Generate CAD-Drawing (F10)
OF ?Fbutton9              !Delete current input (<Ctrl><Delete>)
```

When you press the Source button you'll see that these are nicely aligned in the CASE ACCEPTED loop, under the last control (?Help in my case).

Summary

On other, less complex, windows in this app, where I need to have more than one FileDropBox *on the same file* (and I have plenty of them, with two to six lookups) I only create one FileDropBox and also use my floating FileDropBox-technique.

Note: when your window has Tabs then you *must* populate the FileDropBox on the same tab otherwise it will not hide when the active tab changes!

Addendum

Although not specifically related to the techniques I've described here, I did want to show the output from this screen. Figure 3 shows a Clarion-created schematic.

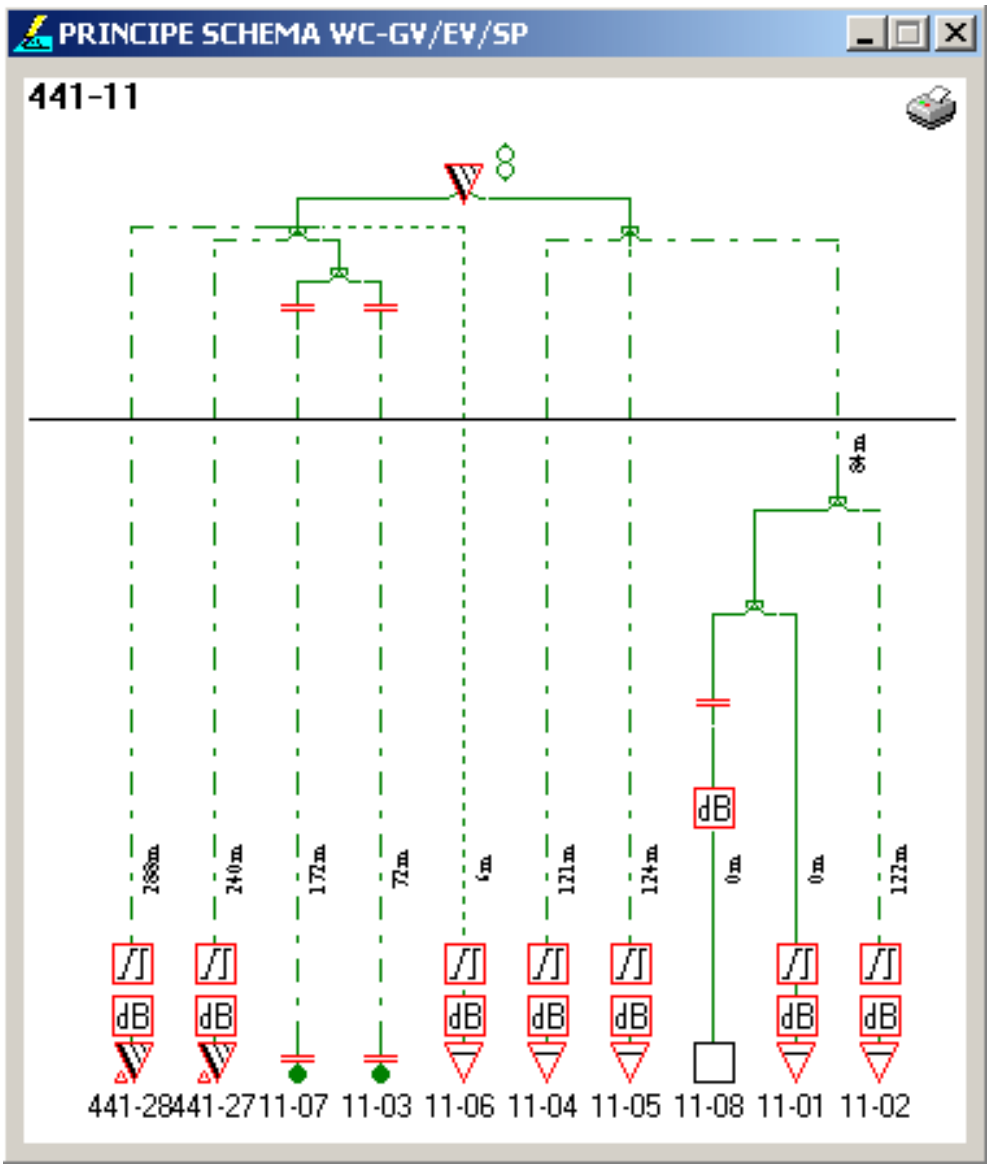

Figure 3. Schematic output created in Clarion

If you look at the input screen in Figure 2, you'll see the similarities within the entered columns and the drawing. On this screen, when the user hovers above any of the elements/cables, a tooltip appears with all info about the element under the mouse-pointer. Alas, my screen capture utility didn't capture the tooltip.

You can view a more complete embedded source listing here.

Maarten Veenstra began his computing career as a field service engineer on the DEC PDP11 family and a microprocessor-controlled punchcard machine. His first personal computer job was fixing CP/M computers and the early IBM clones, during which time he bought an MSX "computer" and taught himself BASIC and assembler. Maarten began using Clarion

Professional Developer in 1988, wrote his first Windows program with CW 2.0, and his first ABC program with C4. He is now the co-owner of Nepucon, which provides service to public utilities. Maarten married in 1991, and he and his wife have adopted two beautiful Chinese girls.

Reader Comments

Add a comment

Clarion Magazine

Internationalization Tools Standards: Learning from an ABC Calendar Workaround

by Phil Will

Published 2005-10-20

Anyone who has internationalized a large application knows that one of the more time consuming tasks is dealing with third party products which attack internationalization in ways that do not allow languages choices at run time. Even in a pure ABC context, creating multi-language applications is challenging but generally possible, as the translator class is included, by reference, in the window manager and report manager class and in some of the other classes.

Two notable exceptions within the ABC library are the Calendar and List Format Classes introduced with Clarion 6. Although I use PD 1-Touch Date Tools Calendar Classes which are fully internationalized, I thought it might be instructive to look at the class and how it could be modified, both as an aid to someone who might be committed to its use, and as an illustration of an approach to internationalization that could be used as a starting model by third party tool developers.

Before getting into the ABC Calendar Class, I would suggest the following general standards for internationalization ready tool design.

1. **User Interface Elements**. All user interface elements and default values should be located in a translation file (file with a .trn extension). These files, introduced in Clarion 4, can be copied to an application directory and customized by developers as needed. Because these files all have a trn extension, they can be easily identified. If you are using a parsing utility such as the Translator Plus Source Manager, a common extension makes it easy to add these files to the list of source files to be parsed.
    a. **Windows.** If there are one or more windows used by a class, these should be referenced by section in the translation file.
    b. **Default Strings.** If the tool is being used by applications developed in languages other than English, this is a place where developers can change the defaults to their own language, i.e. a single language translation. This is typically done with equates for each source string. Other defaults such as icons or specific properties can also be included.
    c. **Source/Replacement Map Structure.** It can be helpful to include a section containing source and replacement strings in a translation map format. I would consider this highly desirable but not absolutely necessary, since the translator class has the ability to export strings if needed (unless the strings are not easily displayed and therefore not easily exported).
2. **Multi-Language Translation.** The library should allow for multi-language, run time translation (as opposed to a one-time, static translation into another language). There are a couple of options

for handling this.

a. **TranslatorClass Reference.** In this case, the `TranslatorClass` can be declared as a reference property within the class and used to do translations. This is comparable to the technique used for the `WindowManager` class. This would require an `AddItem(TranslatorClass pTranslator)` method and implementation of the method if runtime translation is turned on.
b. **Translation Method.** Add one or more virtual methods for translation; this can easily be used with a wrapper template that conditionally generates translation calls if run time translation is enabled (`%EnableRunTimeTranslator=%True`). The methods called would typically be the `TranslatorClass`'s `TranslateWindow`, `TranslateString`, and `TranslateControl` methods.

ABC Calendar Class

The ABC Calendar class as implemented in the shipping version of Clarion 6 consists of a class library and control template for creating a popup calendar button. The popup calendar can be displayed as a small popup calendar (`CalendarClass`), or as an expanded calendar with several date manipulation buttons (`CalendarSmallClass`). The class declarations are in ABUTIL.INC; the implementing code is in ABUTIL.CLW; and the interface elements in ABUTILUI.CLW (a violation of the translation file naming convention above). The interface file contains window declarations and default strings; there is no source/replacement translation group structure and, if you examine the code, no way to translate the window into multiple languages at run time.

The class has several undocumented properties that can be changed at run time, among them colors, whether a date is selected on close, X and Y positions, alignment, and the first week day in the calendar display. The last is important from an internationalization perspective. The class also has an `IsHoliday` virtual method that could be used to display a holiday in a different color.

Creating workarounds for tools that fail to provide for multi-language translation can involve code modification, template modification, or both. In this case, where all the code is source code and class based, it seemed to me that the creation one or more derived classes might be the easiest approach. Further examination of the SV calendar classes reveals the following:

1. **CalendarBaseClass.** Both calendars share a base class containing most of the code. Most class libraries consist of small blocks of code which allow you many opportunities to modify behavior. This was not done here.
2. **Ask Method.** The call to the calendar is handled by the `Ask` method, which is a large block of code that contains the `ACCEPT` loop and does all of the calendar handling except for opening the window. The calendar windows are opened in the derived `Ask` methods, one for each calendar type.
3. **Setup Method.** The base class has a virtual `setup` method where you can make changes to the customizable properties. The `TranslatorClass` can translate the last opened window without a direct reference to the window (the window parameter of `TranslateWindow` can be omitted). Since the call to this method occurs after the window is opened, at first glance it would appear that the window could be translated within this method. However, this call occurs before day and month names and the calendar title are assigned.
4. **Overlay Groups and Arrays.** Looking at the code also reveals that abbreviated day names and month names are handled using `STATIC GROUP`s with day names (declared in ABUTILUI.CLW and shown in Listing 1), which are then included by `SECTION` reference in the

base class `Ask` method data with an associated array as shown in Listing 2. These inaccessible names require translation at run time.

Listing 1. Groups declaring default day and month names

```
SECTION('CalendarDayGroup')
Day_group            GROUP ,STATIC
d1                   STRING('Mon')
d2                   STRING('Tue')
d3                   STRING('Wed')
d4                   STRING('Thu')
d5                   STRING('Fri')
d6                   STRING('Sat')
d7                   STRING('Sun')
                     END
!!
!! Month Group declaration with Day labels
!!
 SECTION('CalendarMonthGroup')
Month_group          GROUP ,STATIC
m1                   STRING('January   ')
m2                   STRING('February  ')
m3                   STRING('March     ')
m4                   STRING('April     ')
m5                   STRING('May       ')
m6                   STRING('June      ')
m7                   STRING('July      ')
m8                   STRING('August    ')
m9                   STRING('September ')
m10                  STRING('October   ')
m11                  STRING('November  ')
m12                  STRING('December  ')
                    END
```

Listing 2. Included Group with Overlaid Array

```
INCLUDE('ABUTILUI.INC', 'CalendarDayGroup')
Day_array            STRING(3),DIM(7),OVER(day_group)
 INCLUDE('ABUTILUI.INC', 'CalendarMonthGroup')
month_array          STRING(10),DIM(12),OVER(month_group)
```

Multi-language class workaround

To add multi-language translation, I decided to modify the base class to provide for a translation hook. I also modified the derived classes for each of the calendar types so that their parent calls would utilize the new base class. I added a `TranslatorClass` reference property, and two methods: `Translate(*STRING[] pMonthArray,*STRING[] pDayArray)` to do the actual translation of both the window and the arrays, and `AddItem(TranslatorClass pTranslator)` to assign the `TranslatorClass` reference property.

The definition file is shown in Listing 3 below (some line breaks added). Those familiar with creating ABC

Compliant Classes will recognized the `!ABCincludeFile` which is a flag for including the class definition in class ABC Class Information used by the templates. I did not use a family name parameter (`!ABCIncludeFile[FamilyName]`) because I did not anticipate creating a template for this class that would include the library by reference only. Generally when creating a third party template I would include the family name to avoid forcing the code to compile in a multi-DLL ABC application or Clarion chain application using with ABC classes turned on.

Listing 3. The definition file

```
!ABCIncludeFile
 OMIT('_EndOfInclude_',_PDUtilitiesPresent_)
_PDUtilitiesPresent_ EQUATE(1)
  INCLUDE('ABUTIL.INC'),ONCE
PDCalendarBaseClass  CLASS(CalendarBaseClass),TYPE,|
                         MODULE('PDABUTIL.CLW'),LINK('PDABUTIL.CLW',|
                         _ABCLinkMode_),DLL(_ABCDllMode_)
Translator             &TranslatorClass
Ask                    PROCEDURE(STRING pTitle,LONG pDate=0),|
                         LONG,PROC,DERIVED
Translate              PROCEDURE(*STRING[] pMonthArray, |
                         *STRING[] pDayArray),VIRTUAL
AddItem                PROCEDURE(TranslatorClass pTranslator),VIRTUAL
                     END
PDCalendarClass      CLASS(PDCalendarBaseClass),TYPE,|
                         MODULE('PDABUTIL.CLW'),LINK('PDABUTIL.CLW',|
                         _ABCLinkMode_),DLL(_ABCDllMode_)
Ask                    PROCEDURE(STRING pTitle,LONG pDate=0),LONG,PROC,DERIVED
                     END
PDCalendarSmallClass CLASS(PDCalendarBaseClass),TYPE,MODULE('PDABUTIL.CLW'),|
                         LINK('PDABUTIL.CLW',_ABCLinkMode_),DLL(_ABCDllMode_)
Ask                    PROCEDURE(STRING pTitle,LONG pDate=0),LONG,PROC,DERIVED
                     END

!_EndOfInclude_
```

The `PDCalendarBaseClass.Ask` method is an exact copy of the SV code except for the addition of the call to the `translate` method. I also moved the assignment of the window title to an earlier position in the code to allow for the translation of the window title as well.

Listing 4 . Base class Ask method changes

```
  . . .
  IF CLIP(pTitle) THEN    !-- Move to before translation
     0{prop:text}=pTitle
  ELSE
     0{prop:text}=''
  END
  SELF.Translate(Month_Array,Day_Array)  !-- Added call to translation.
  DO PrepareProcedure
  lHIni=0{PROP:HEIGHT}
!  IF CLIP(pTitle) THEN
!     0{prop:text}=pTitle
```

```
!  ELSE
!     0{prop:text}=''
!  END
  DISPLAY()
  lHEnd=0{PROP:HEIGHT}
  IF lHIni<>lHEnd THEN
     0{PROP:HEIGHT}=0{PROP:HEIGHT}+(lHIni-lHEnd)
  END
  SELF.Response = RequestCancelled
  ACCEPT
    . . .
```

The new base class `AddItem` method simply assigns the `TranslatorClass` to the translator property. The `Translate` method translates each of the arrays and then calls the `Translator.TranslateWindow` method which will translate everything else in the window. The translator methods are not used if the `TranslatorClass` has not been assigned.

Listing 5 . AddItem and Translate methods

```
!-------------------------------------
PDCalendarBaseClass.AddItem PROCEDURE(TranslatorClass |
                             pTranslator)
  CODE
  SELF.Translator &= pTranslator
!-------------------------------------
PDCalendarBaseClass.Translate PROCEDURE(*STRING[] pMonthArray,|
                                 *STRING[] pDayArray)
  CODE
  IF ~SELF.Translator &= NULL
    !-- Translate Month Names
    LOOP I#=1 TO 12
      pMonthArray[I#]=|
          SELF.Translator.TranslateString(CLIP(pMonthArray[I#]))
          !-- Note CLIP is needed here.
    END
    !-- Translate Day Names
    LOOP I#=1 TO 7
      pDayArray[I#]=|
        SELF.Translator.TranslateString(CLIP(pDayArray[I#]))
    END
    !-- Translate buttons etc.
    SELF.Translator.TranslateWindow()
  END
```

The class code for each of the derived calendars is exactly the same as the respective SV calendar classes; the only difference is that they are derived from the new base class, so they will call the new ask method.

Applying the multi-language ABC calendar

To develop and test the library, I created a simple application with a file and two dates, one for each of the

calendar types. I also created a translation file (PDABCCal.trn) with British English, French, and German translations, a user language preference procedure, and a procedure to load selected translations

Translation file (PDABCCal.trn)

The translation file consists of group structures with source and replacement strings for each translation. The structure is the same as that used by the TranslatorClass library. While there are other ways to load translations which make customization and maintenance easier, this serves well for illustrative purposes. The translation file is located in the application directory

User interface file (ABUtilui.clw)

Since I am programming in United States English, on the surface there was no need to change the window definitions or the day and month names in the user interface file. Note that SV chose to use a CLW file rather than a translation file; because this is referenced in the shipping ABC class libraries, I chose not to change the extension. In testing, however, I found that there was a `STATIC` attribute on the group declarations that made it impossible to change the language once a calendar had been translated. This attribute is commented out. I would also suggest changing date and time pictures from `@d2` and `@t6` to `@d17` and `@T7` which will pick up the user's Control Panel short date and time pictures by default. The changes were made to a copy of the file located in the application directory.

Sample application

The sample application has the following elements that are an important part of an application with translation capability:

1. Enable Run-Time Translation is checked.
2. On the Global Classes tab, Translator Configure button, Additional Translation Groups Button, no groups are identified. For the multi-language translation, this is hand coded to allow selection of the group to be added. Note that if you do enter a group, you can include a section reference by adding ',' after the file name before the section; the template will add the single quotes to either end and double the entered quotes.
3. Global fields have been added for a user selected language and the first week day to display on the calendar.
4. A preferences procedure has been added to allow the user to select the language. The selection is saved using the `IniMgr`.
5. An `AddLanguage` procedure adds translations used by the `TranslatorClass` and sets the first week day value in accordance with the selected language.

Preferences

The Calendar Preferences window in Figure 1 allows users to select a language. If the OK button is pressed, the language selection is saved by the `IniMgr` and loaded using the `AddLanguage` procedure.

Figure 1. Calendar preferences

The language entry options are specified as local data with a Must be in List entry, as in Figure 2. Values are standard three character language codes except for the first default entry.

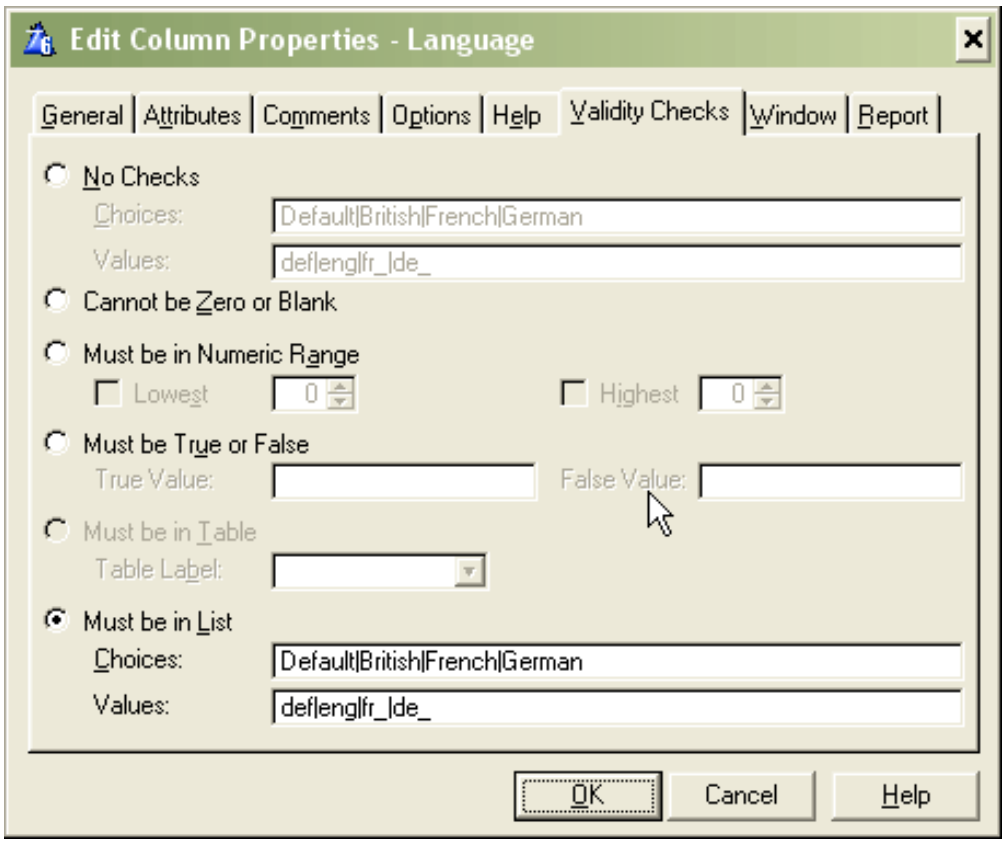

Figure 2. Setting the validity checks

Listing 7 . AddLanguage procedure init code

```
IniMgr.Fetch('Setup','Language',Language)
IF Language='' then Language='def'.
SELF.Okcontrol=?OK
```

Listing 8 . AddLanguage TakeCompleted code

```
IniMgr.update('Setup','Language',Language)
gLanguage=Language
AddTranslation
! Parent Call
ReturnValue = PARENT.TakeCompleted()
! [Priority 7500]
POST(EVENT:CloseWindow)
```

The AddLanguage procedure

The AddLanguage procedure includes the .trn file in its data section and the uses the `TranslatorClass`'s `AddTranslation` method to load the translations for the selected language.

Listing 9. The AddLanguage procedure

```
  INCLUDE('PDABCAL.TRN','GROUPS'),ONCE
! End of "Data Section"
  CODE
! Start of "Add additional DebugHook statements"
! [Priority 5000]
```

The code section uses the `TranslatorClass`'s `AddTranslation` method to load source and replacement strings. By default, this replaces any existing translation for a given source string, in effect making the new translation override any prior translation. The case structure uses the current global language value and also assigns the related first week day to a global variable.

```
CASE gLanguage
OF 'Def' OROF ''
  gFirstWeekDay=7
  Translator.AddTranslation(PDCalDef)
OF 'eng'
  gFirstWeekDay=1
  Translator.AddTranslation(PDCalEng)
OF 'fr_'
  gFirstWeekDay=1
  Translator.AddTranslation(PDCalFr_)
OF 'de_'
  gFirstWeekDay=1
  Translator.AddTranslation(PDCalDe_)
END
```

DateForm

Figure 3 shows the date entry form.

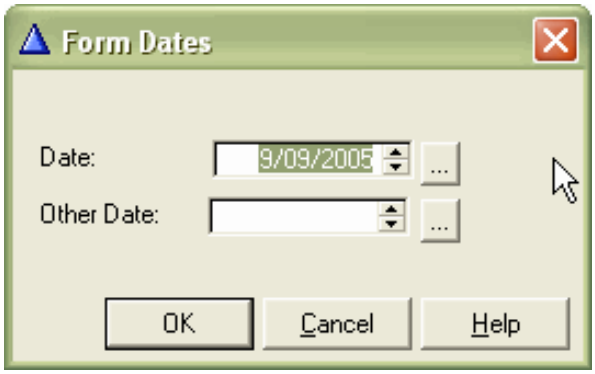

Figure 3. The date entry form

The Date entry form displays the two date fields, each using one of the two multi-language calendar button options.

In addition to filling out the normal template prompts on the General tab, the following steps are needed to use the new classes (see Figure 4):

1. On each of the calendar classes tab, the class needs to be change from the default class to the newly defined PDCalendarClass or PDCalendarSmallClass. Uncheck the UseDefault ABC CalendarClass, check Use Application Builder Class, and select the appropriate class.
2. Each calendar button requires two lines of embed code – one adding the translator to the class and one to set the first week day to display on the calendar.

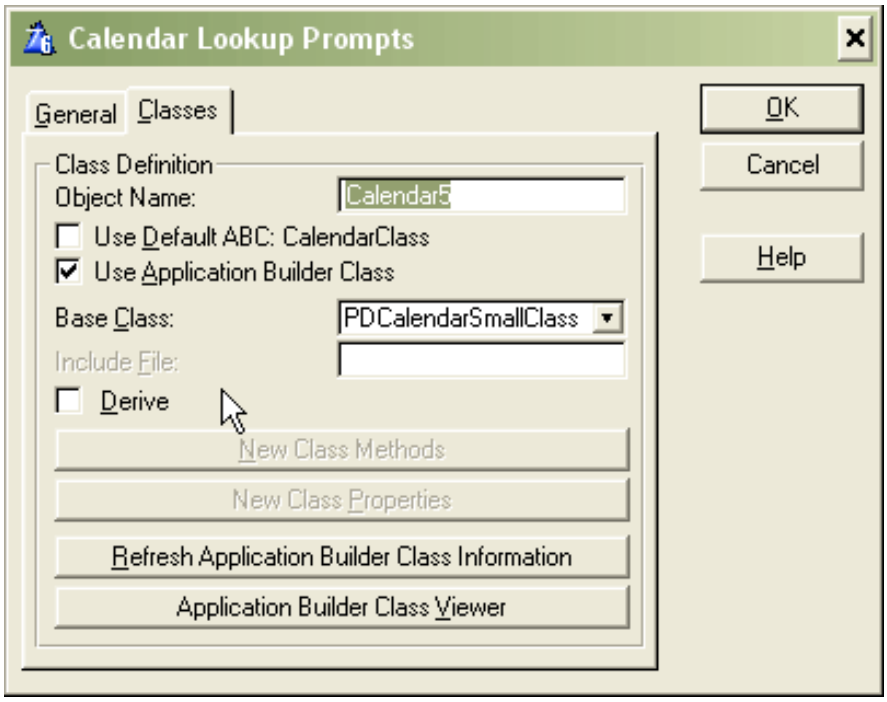

Figure 4. Setting the Calendar Lookup Prompts

You could code your own control template to do this for you or write a workaround extension template that populated the code for all calendar buttons. Note also that the calendar title is passed as parameter to the `Ask` method. If you want the title translated, be sure to include its translation in the translation group structures. My own preference is to have no title, and have the calendar pop up below or above the date control so that it is clear what is being entered. I've found it best to position the calendar using pixels, whose coding is beyond the scope of

this article.

Listing 10. Hand coded additions

```
ThisWindow.Init
Calendar5.AddItem(Translator)
Calendar8.AddItem(Translator)
```

Here's the code for the calendar setup method:

```
SELF.FirstWeekDay=|
   CHOOSE(~gFirstWeekDay,7,gFirstWeekDay) !-- BYTE(7) Sun=7
```

Figures 5 and 6 show the running calendars translated to German

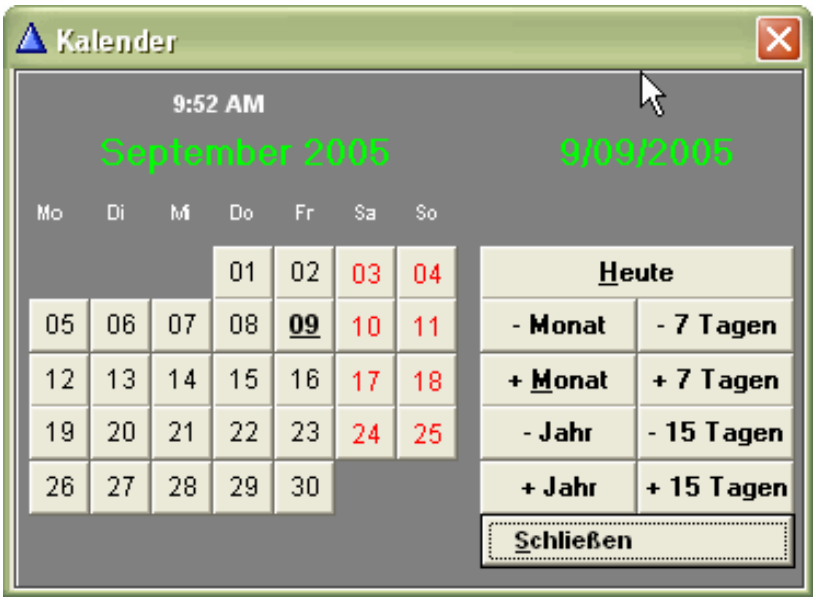

Figure 5. The large calendar in German

Figure 6. The small calendar in German

Conclusion

The ABC Calendar as shipped does not meet the suggested standard for international tool design in two respects: 1) the user interface elements are not in a translation file (.trn file), and 2) there is no way to implement a multi-language application without modification. Using derived/replacement classes, developing a workaround is possible but is something that should not be necessary. One still also has to do some hand coding for each popup which is an unnecessary time waster. That fact the ABC Calendar has an interface file was helpful in that it allowed the removal of the group static attributes. It was, however, missing a translator group structure with source strings that could be used for creating translations.

If I were developing an international application, I would look for a third party tool that handles internationalization more easily. The specs would include accurate positioning of the calendar (minimally handled by the ABC Calendar), use of multi-language registry information for day and month names and first week day, handling of range limits, handling of days with scheduled items as well as holidays, quicken keys, advanced jump to options, and, if you really want RAD development, run time creation of calendar buttons for all date entries by the simple addition of a global extension.

If you download and use the code from this article, be sure to put the PDABUTIL.INC, CLW, and TRN files in the libsrc directory and refresh the ABC Application Builder Class Information.

Download the source

Philip S. Will is President of ProDomus, Inc, a SoftVelocity Third Party Accessories Partner and Clarion applications developer. He has been a presenter at several Clarion conferences, and has published articles on Internationalization and template writing. His principal third party products include PD Lookups, PD Translator Plus, and PD 1-Touch Date Tools. Philip has been coding in Clarion since 1991.

Reader Comments

Add a comment

Clarion Magazine

A Tree in a Page Loaded Browse: the Sequel, Part 1

by David Podger and Deon Canyon

Published 2005-10-27

In May 2003 Clarion Magazine published the first of two articles by Ronald van Raaphorst entitled A Tree in a Page Loaded Browse. The two articles spelled out a simple implementation method that intrigued us.

As long ago as September, 1997 one of us (David) had written an article for Clarion Online which, when you boiled it down, was no more than a plea. The article, A Tree in One File begged someone to write a simple solution to the problem of storing a tree in one Clarion .tps file. David was delighted when the then editor of Clarion Online responded by promising some code, and downcast when he backed out, but couldn't blame him. The solution Tom Moseley evidently had in mind was file-loaded, but a page-loaded solution was our pressing need at the time.

There followed an interlude during which we bought and used a commercial product, which solved the problem of the tree-in-one-file by using recursion. Unfortunately, the up and down buttons did not reliably work in this product, when applied to a recursive table. These buttons may have been fixed since. A bug hidden in an object is frustrating. If he wishes, the object's author can remedy it, if not then the bug remains, in practice, unfixed. We needed a solution that was under our control.

Before venturing further, we need to explain what "up" and "down" buttons do, and why they matter so much in a tree file. Consider the fragment of a tree shown in Figure 1.

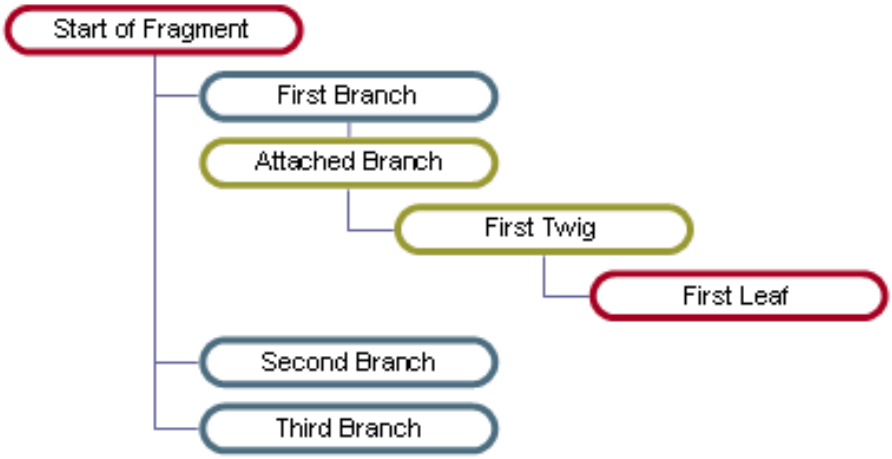

Figure 1. Tree Fragment

Suppose that, in this figure, the First Branch is highlighted and the down button is pressed. The desired result is obvious: the new order of the tree should be:

- Second Branch
- First Branch
- Third Branch

Everything presently attached to First Branch should remain attached to it. In other words, when a node in a tree moves, all its child nodes must move with it. If they do not, there is a *big* problem. A tree structure imposes special conditions on other standard processes. If a node is deleted, for instance, then something sensible must be done with its children.

Ronald van Raaphorst began his first article with a thorough examination of Clarion's template for a tree list control, the Relation Tree. As he said, "one of the disadvantages of this control is that the maximum tree level is fixed by the number of files you use in it." He then went on to describe, in his own words: a "very simple, basic idea for a non-level-limited, page loaded tree, using a standard browse control". What was immediately appealing, on top of his finding a way to use the standard page loaded browse, was that he made no use of recursion. His code was easy to understand and we could change it and add to it. It was, however, incomplete. A planned third article, which was to have published more code, did not appear. Well, Ronald had given us enough (for which, many thanks to him); we could do the rest.

There was in fact another drawback in his solution, but it did not matter. The solution van Raaphorst describes as "non-level-limited" is not *literally* unlimited. Each level adds around five characters to a controlling string.,If there is a maximum of 50 levels, then the control string must be 250 characters in length, if 100 levels, then 500 characters, and so on. But our tree was going to be visible in a Clarion browse window. Each new level therefore meant another indent in the tree. We would run out of usable, visible space in the browse before we ran out of control string characters. It was a non-problem for our intended use of it.

Before you read on, please take a detour now and look at A Tree In A Page Loaded Browse and A Tree

In A Page Loaded Browse: Update And Delete Logic. What follows will assume you have read them, or at least skimmed through.

Using a Tree File to represent a dialogue

There are many uses for a tree file, but our interest is in using it to represent dialogue. This will be our example throughout. A bare-bones example of a dialogue tree is shown in Figure 2.

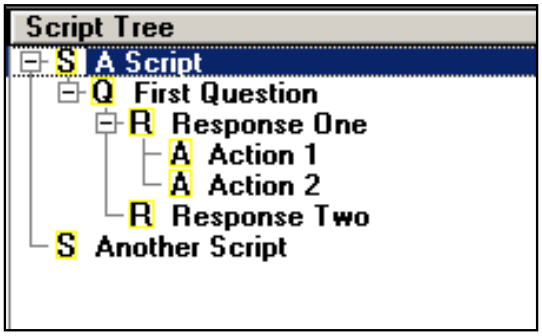

Figure 2. A simple dialogue tree

There are four record types shown in Figure 2:

- **S** Script - a script consists of a set of questions
- **Q** Question - one or more possible responses are attached to each question
- **R** Response - actions are executed when a given response is chosen
- **A** Action - an action executes an expression and saves the result

At run-time, when a script is executed, the user chooses a pathway through the script by selecting from amongst the offered responses. We are not concerned any further in this article with what happens at run-time. Our focus is on design-time and how a tree structure can represent such a dialogue.

As well as the above, follow-on questions may be attached to responses. Figure 3 shows an example.

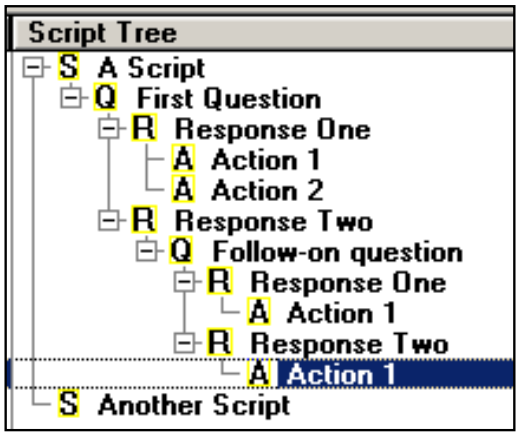

Figure 3. Follow-on Questions

Further questions (and their associated responses and actions) may be added, to the depth allowed by the

control string mentioned above.

It will help at this point briefly to explain Ronald van Raaphorst's contribution to the design of a tree file. He proposed that for each entry in the above tree there should be a hidden `CSTRING` field that controlled everything. In the code below we use his name (`SeqNo`) for this field. It is the only additional data field needed to produce the above tree in a standard page loaded browse.

Here are the control strings for the entries in Figure 3:

```
Type  Control string                 Level
S     0001                          1
Q     0001.0001                     2
R     0001.0001.0001                3
A     0001.0001.0001.0001           4
A     0001.0001.0001.0002           4
R     0001.0001.0002                3
Q     0001.0001.0002.0001           4
R     0001.0001.0002.0001.0001      5
A     0001.0001.0002.0001.0001.0001 6
R     0001.0001.0002.0001.0002      5
A     0001.0001.0002.0001.0002.0001 6
S     0002                          1
```

Each additional level in the tree adds another five characters to the control string.

The tree file is kept in the sort order dictated by this string. A brief inspection of the above control strings will confirm that they are indeed in that order. The decimal points that separate each level are for readability only, but very handy for that, as can be seen. When it comes to inserting additional entries anywhere in the tree, there is no problem, there is room for thousands of them. Were more needed, then five digits could be used instead of four, giving a capacity of 99,999 records *at each level* (with six bytes needed for each level).

What about the Level values shown above, where do they come from? As explained, the control string provides everything. It gives the level number, by using this simple expression: `Level = INT( LEN(SeqNo) / 5 ) + 1`

As explained in van Raaphorst's first article, when a browse is made into a tree by ticking the Tree checkbox, as shown in his diagram (copied here as Figure 4), the template adds an additional field to the browse queue.

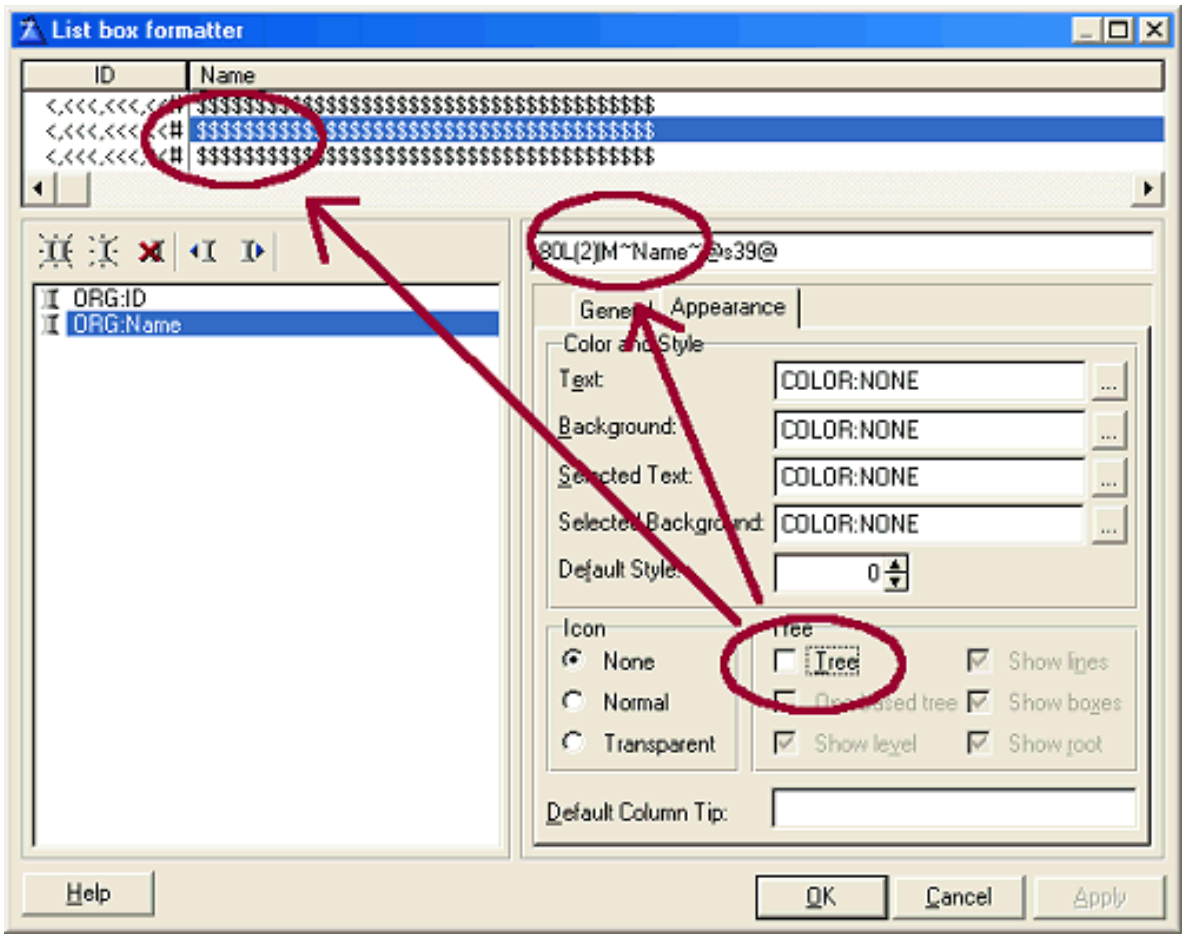

Figure 4. List Box Formatter

This template-generated field holds the level value. Its contents always contain the correct level number because of a single line of embedded code, as shown in Figure 5.

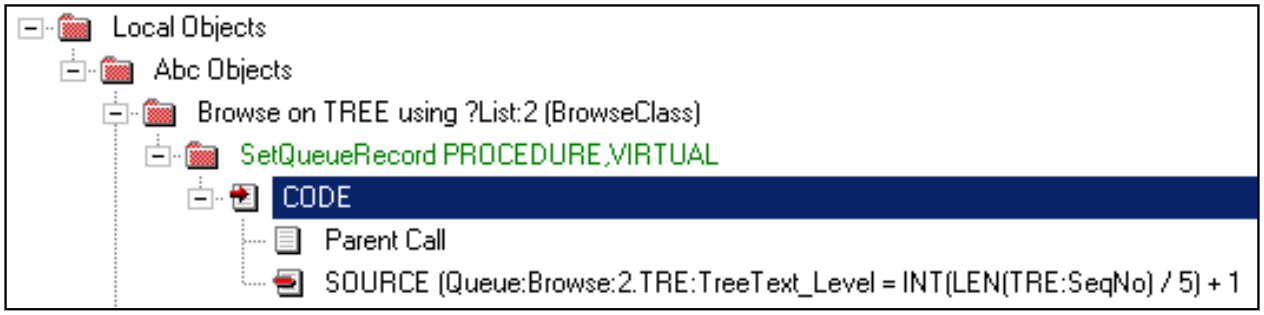

Figure 5. Calculating the Level

In our example, the extra field is named `Queue:Browse:2.TRE:TreeText_Level`. The field is made equal to the correct tree level with this one line of code:

```
Queue:Browse:2.TRE:TreeText_Level = INT(LEN(TRE:SeqNo)/5)+1
```

A reminder, the generated field is in the browse queue, not in the file.

Preparing the ground to begin coding

Our dialogues are termed Plays. Each Play consists of a number of Scripts. The records in the `Play` file are auto-numbered, as are all the records in the `Tree` file, which contains the Scripts for the Plays. A bare-bones dictionary for `Tree` is:

```
TRE:Key_Auto              ! auto values never change
    PlayAuto
    TreeAuto
TRE:Key_SeqNo             ! this key keeps the tree in order
    PlayAuto
    SeqNo
PlayAuto   LONG
TreeAuto   LONG
TreeType   STRING(9)      ! String, Question, Response, Action
TreeText   STRING(1000)   ! common to all tree types
TreeTag    STRING(1)      ! turn copy (C) or cut (X) icon on
SeqNo      CSTRING(255)   ! the control string
```

Because we have at least four different record types (Script, Question, Response and Action), it follows that there will be additional fields that are particular to each type of record. These extra fields are held in separate files, each one in a One-to-One relation to the `Tree` file. `Key_Auto` (made up of `PlayAuto` and `TreeAuto`) is used to join these supplementary files to the `Tree` file. Obviously, once allocated by auto-numbering, `PlayAuto` and `TreeAuto` are never changed, so these joins are permanent and can be used for cascaded deletes and so on. In this article we are not concerned any further with these supplementary files as our focus is on the `Tree` file and its representation of a dialogue's structure.

The local dictionary is as follows:

```
LenSeq          LONG          ! length of TRE:SeqNo
BasePlay        LONG          ! copy of TRE:PlayAuto
BaseSeq         CSTRING(255)  !    "    TRE:SeqNo
NewSeq          CSTRING(255)  !    "    TRE:SeqNo
ChildSeq        CSTRING(255)  !    "    TRE:SeqNo
SaveSeq         CSTRING(255)  !    "    TRE:SeqNo
CascadeLevel    STRING(4)     !    "       "
Direction       STRING(4)     ! 'Up' or 'Down'
TreeLevel       BYTE
ShowType        STRING(10)
CutOrCopy       STRING(1)     ! 'X' or 'C'
CutPlay         LONG          ! saves a cut entry
CutLevel        LONG
CutSeq          CSTRING(255)
CutType         STRING(9)
BumpPasteChild  LONG          ! paste info
PasteLevel      LONG
SaveTreeAuto    LONG          ! copy of TRE:TreeAuto
NoTree         BYTE           ! TRUE if empty Tree
```

Two queues are used; one acts as a cut buffer and the other makes coding simpler when manipulating the `Tree` file:

```
CutQu       QUEUE,PRE(CUT)
              LIKE(TRE:Record)
            END
TreeQu      QUEUE,PRE(TQU)
              LIKE(TRE:Record)
            END
```

In the Default Behaviour tag for the `Tree` browse (see Figure 6), entries are made to limit the display to the Trees for one Play.

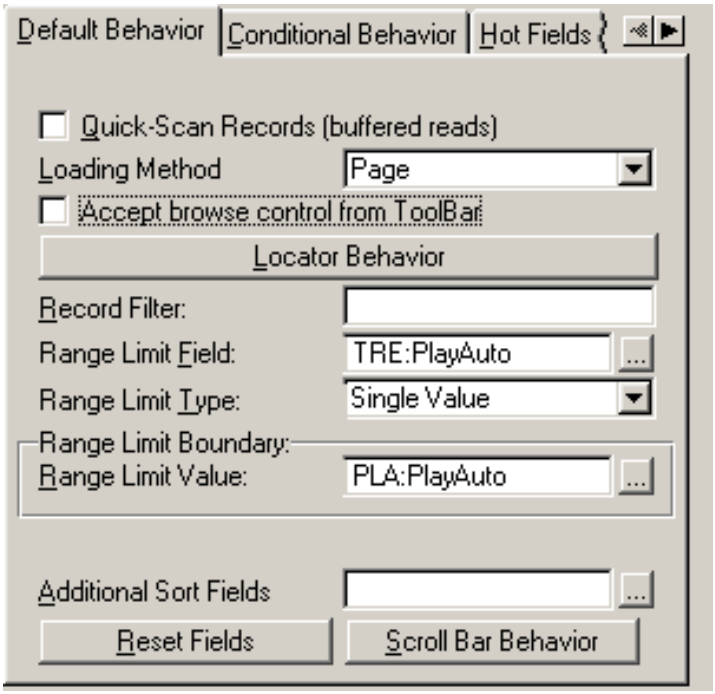

Figure 6. Limiting the displayed Trees to one Play

Icons are defined for the tree. In the browse of Conditional Icons (see Figure 7), three icons are defined for each record type. For Action, these are:

- A.ICO
- ARed.ICO
- AGreen.ICO

A.ICO is an icon showing the letter **A** in black and bold. The next two show the same bold letter, but colored red and green, respectively. The same pattern of three icons is repeated for Question, Response and Script (e.g. Q.ICO, QRed.ICO, QGreen.ICO and so on). The field `TRE:TreeTag` is made blank, X or C to display one of the three icons.

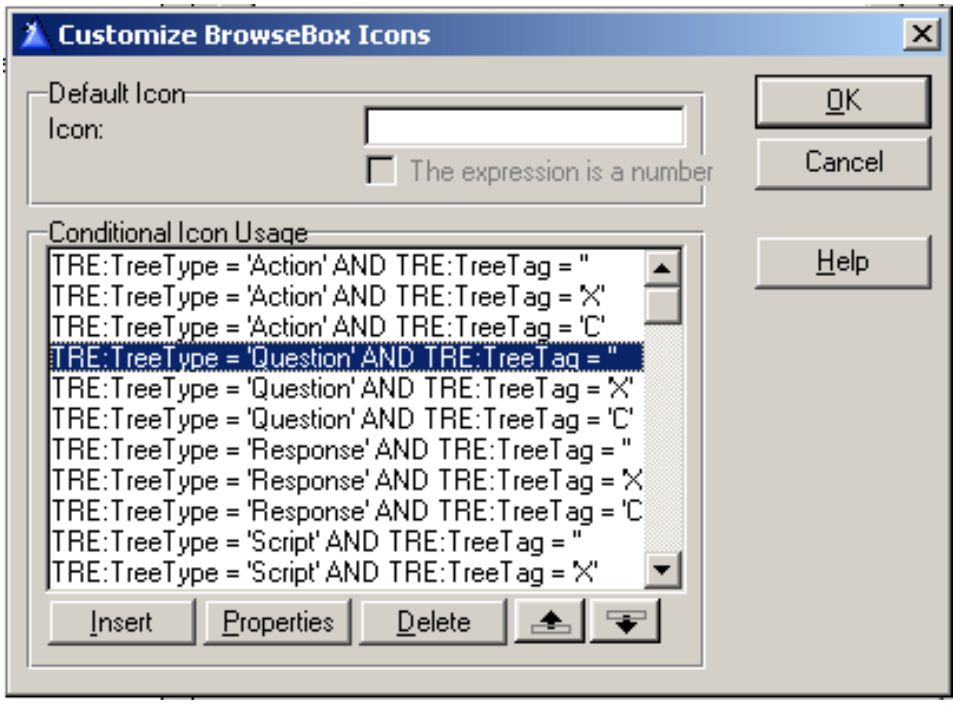

Figure 7. Customised BrowseBox Icons

When a segment of the tree is cut or selected for copying, all icons in the segment change to the appropriate color. When a paste is executed or cancelled, the segment's icons revert to plain black. The code to do this appears later.

The correct handling of inserts is fundamental to the correct coding of a tree file. Unlike an insert into a normal browse, where the contents of one or more fields automatically determine the record's position, an insert into the browse of a tree depends on *which existing record is highlighted at the time*. Based on this, hand-code is required to generate a control string and place it in the `TRE:SeqNo` field, so as to force the positioning of the inserted record.

The basic rule is that the inserted record appears below the highlighted record, or, more strictly, that it inherits the control string of the highlighted record and appends another level to it. Thus, in the simplest case, where the control string of a highlighted record is `0001.0003.0004`, and there are presently no records attached below this record, the inserted record's control string becomes `0001.0003.0004.0001`. If there are already records occupying this position, then the inserted record is appended. For example, suppose there is *already* a record with a control string of `0001.0003.0004.0001` then the control string of the inserted record becomes `0001.0003.0004.0002`. This gives the inserted record a place in the browse at the end of the records attached to the highlighted record.

From the above it will now be clear why the up and down buttons are important in a tree file, and why there is gnashing of teeth when they do not work perfectly. After inserting a new record the user will very often immediately adjust its position (from an initial location at the end of those already present) by nudging it with the up button. Since the newly inserted record will now be highlighted, the user will appreciate doing this using an alerted key combination like Ctrl+uparrow. To accomplish this, CtrlUp and CtrlDown are alerted and the following code is inserted at the Window Events/AlertKey embed

point:

```
IF KEYCODE() = CtrlUp
  POST(Event:Accepted,?ButtonUp)  ! press 'up' button'
END
IF KEYCODE() = CtrlDown
  POST(Event:Accepted,?ButtonDown)! press 'down' button
END
```

The insertion of a Response under a Question is an example of the above insertion rule. With a `Question` record highlighted, the only insert button visible is the New Response button (Figure 8).

Figure 8. New Response button

The other buttons are hidden to make sure the user does nothing untoward. The code to hide and unhide the various insertion buttons is provided later.

If a Response record is highlighted, however, the user has two choices. Firstly, an Action may be inserted with the New Action button (Figure 9).

Figure 9. New Action button

But, since follow-on questions may be asked to a considerable depth, another button is also made visible (Figure 10).

Figure 10. New Question button

In both cases, the generated control string is the same and its construction follows the above rule. That is, it inherits the control string of the highlighted record and appends another level to it.

The rules of the dialogue forbid the attachment of records to Actions. So, when an Action record is highlighted, all insert buttons disappear.

Lastly, there is the one case where a true exception to the rule explained above occurs. This is when a Script record is highlighted, because two kinds of insertion must be allowed, either the insertion of a brand new Script or the insertion of a Question below a Script record. The Script record is not below any other, so there is nothing else to do but highlight another Script record and choose from one of the following buttons, as shown in Figure 11.

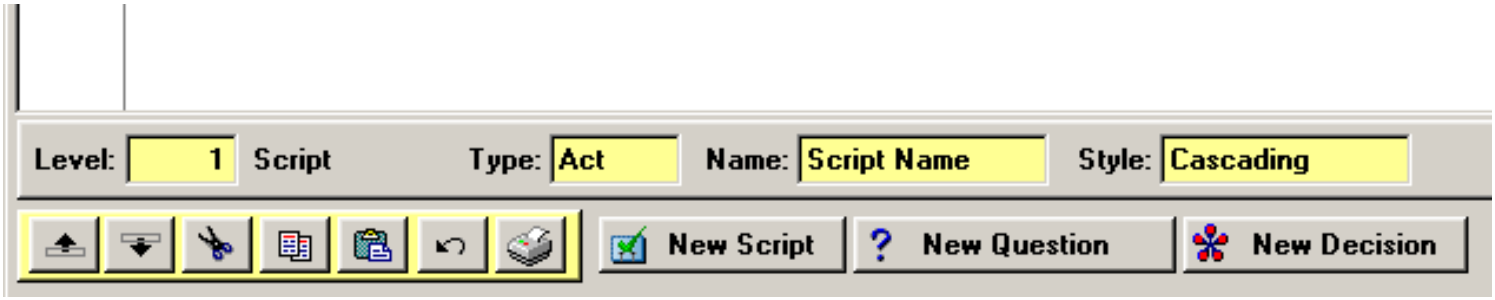

Figure 11. Two insert choices

If the New Script button is pressed the hand-code must generate a control string that makes the new Script the last record in the browse. Suppose there are presently three Scripts in the browse, numbered `0001`, `0002` and `0003`. The new Script is given the number `0004`. The up button may be used, as before, to re-position the whole script relative to the others.

If the New Question button is pressed, then the basic rule given above applies and a Question record is inserted below the highlighted Script.

A special case occurs when the browse is completely empty. In this event, only the New Script button is shown.

To accomplish all this appearing and disappearing of the different insert buttons, there is a routine, `RefreshButtons` which is executed from two embed points:

```
RefreshButtons      ROUTINE
  TreeLevel = INT(LEN(TRE:SeqNo)/5) + 1  ! calculates level
  DISPLAY(?TreeLevel)
  CASE TRE:TreeType
  OF 'Script'
    UNHIDE(?ButtonScript)
    ?ButtonQuestionResponse{PROP:Text} = 'New Question'
    ?ButtonQuestionResponse{PROP:ICON} = 'question.ico'
    ?ButtonQuestionResponse{PROP:TIP}  = 'Add a Question to the Script'
    UNHIDE(?ButtonQuestionResponse)
    SELECT(?TabScript)
  OF 'Question'
    HIDE(?ButtonScript)
    ?ButtonQuestionResponse{PROP:Text} = 'New Response'
    ?ButtonQuestionResponse{PROP:ICON} = 'imptpool.ico'
    ?ButtonQuestionResponse{PROP:TIP}  = 'Add a Response to the Script'
    UNHIDE(?ButtonQuestionResponse)
    HIDE(?ButtonAction)
    SELECT(?TabQuestion)
  OF 'Response'
    HIDE(?ButtonScript)
    ?ButtonQuestionResponse{PROP:Text} = 'New Question'
```

```
    ?ButtonQuestionResponse{PROP:ICON} = 'question.ico'
    ?ButtonQuestionResponse{PROP:TIP}  = 'Add a Question to the Script'
    UNHIDE(?ButtonQuestionResponse)
    SELECT(?TabResponse)
  OF 'Action'
    HIDE(?ButtonScript)
    HIDE(?ButtonQuestionResponse)
    HIDE(?ButtonAction)
    SELECT(?TabAction)
  END
  SELECT(?List:2)
```

To execute the above routine, the following single lines of code are located at the Control Events embed point (`List:2` is the name of the Tree browse), as shown in Figure 12.

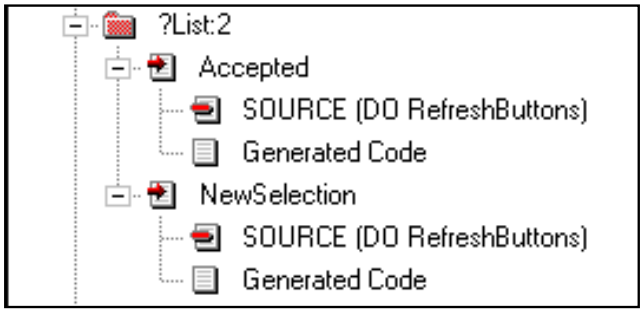

Figure 12. Control Events embed point

When the Tree browse is empty only the New Script button appears, using the embed point shown in Figure 13.

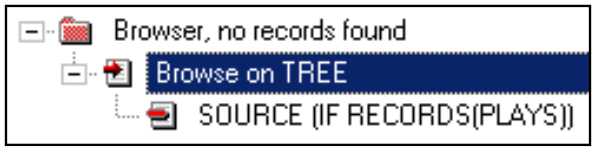

Figure 13. Embed at Browser, no records found

The full code for the embed is:

```
IF RECORDS(PLAYS)
  UNHIDE(?ButtonScript)      ! only show the New Script
ELSE                         ! button if a Play exists
  HIDE(?ButtonScript)
END
HIDE(?ButtonDecisionCondition)
HIDE(?ButtonQuestionResponse)
HIDE(?ButtonAction)
NoTree = 1
```

The NoTree variable is cleared at another embed point, as shown in Figure 14.

Figure 14. Embed at Browser, records found

The effect of the above is to hide and unhide the various insert buttons appropriately as the user moves the highlight bar in the browse or when a row is clicked with the mouse. The `?ButtonQuestionResponse` control has dynamic content which is adjusted as well, so as to avoid having another button.

Next week we'll begin explaining the source code.

Download the source

David Podger has been an independent Clarion developer in Australia for a decade. He presently lives in Sydney, where he sells a specialised accounting application for remote communities.

Dr Deon Canyon is a Senior Lecturer (translation: Associate Professor) in public health and medical entomology at James Cook University in North Queensland, Australia. His current research interests include: Disease control simulation models for use in distance education; *Lymphatic filariasis* (a disease caused by thread-like parasitic worms) and the *Aedes polynesiensis* mosquito in transmission and control of the Pacific Head Lice (*Pediculus capitis*); Health in remote settlements and the use of touch screen kiosks to improve it; Aedes aegypti mosquito behavior, physiology and ecology . Deon is married, and has three children. His hobby of choice is kite surfing.

Reader Comments

Add a comment

Clarion Magazine

Exporting APPs and DCTs to XML

by Harley Jones

Published 2005-10-28

This article is a precursor to another article I plan to write. The second article will be about XSLT techniques for generating code and proprietary file layouts. (XSLT is an XML dialect used for transforming one type of XML into another XML or text-based format.) This article, however, is a demonstration of the advanced Clarion template techniques I used to transform Clarion APP and DCT files into XML. It's actually much easier than you think! And some of those template techniques, including the use of all-important `#CONTEXT` structure, have wide application.

Why XML?

Before I go into how I created XML from APP and DCT files, I should explain *why* I did this. My initial motivation was just to do it. I had finally achieved that "eureka" moment with XSLT, and I honestly didn't know what use I could find by taking the DCT or APP to XML. But it was a challenge, and it was something to do when my brain needed a breather from real work. And as soon as I finished the DCT template, a customer asked for some pretty detailed documentation about the database. Just a coincidence, but fantastic timing! I easily used XSLT to transform the dictionary into HTML documentation (complete with links between keys and their fields, and table relationships). In my opinion, it was much easier than writing a Clarion template to do the same thing. Once I saw that the work actually paid off, I was more inclined to do the APP template too.

The Dictionary

Let's start with the Clarion Dictionary (DCT) file. In the simplest explanation, to export the dictionary in XML I merely looked at the template symbol hierarchy listed in the Online Help. Beginning with `%DictionaryFile` symbol, the template hierarchy contains a very complete mapping of every bit of information a Clarion developer can store within the DCT file. I broke the hierarchy down into logical pieces. This eased my development, and it also allows the user (you) to pick and choose exactly what you want to generate. I ended up with the following categories: Dictionary, File, Key, Field, Relation, and File Driver Info. If the user leaves all the checkbox prompts unchecked, the export will produce a simple XML list of file labels. With everything checked, the export will produce a complete XML file with every bit of detail stored

in the DCT file.

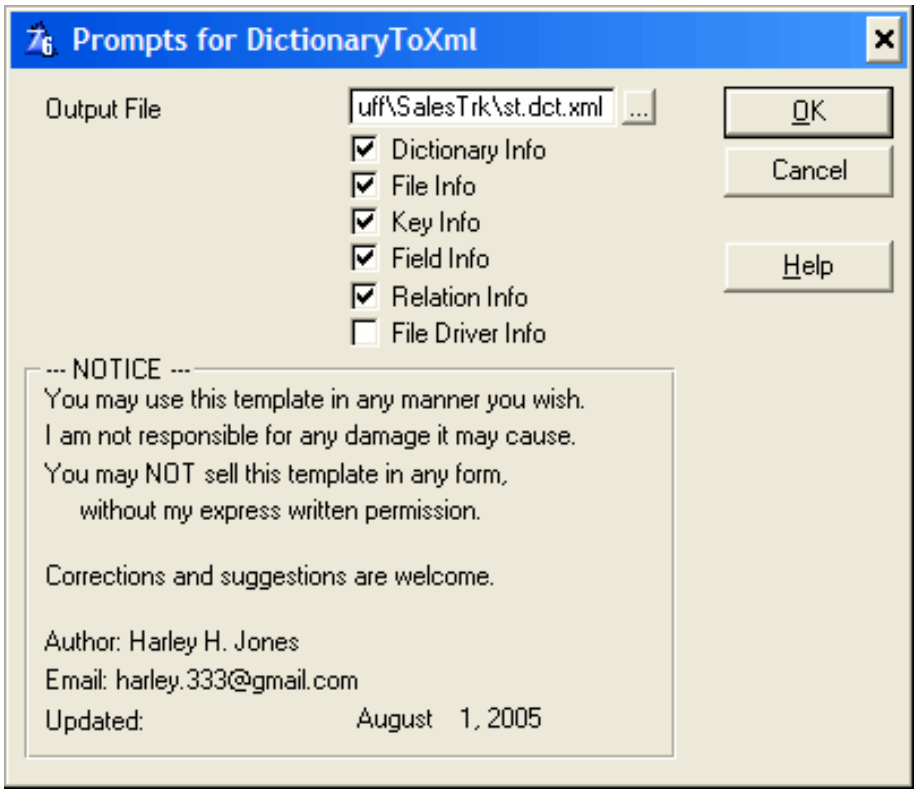

Figure 1. The dictionary export template prompts

Often, template authors create #EXTENSION templates that users can attach to their APP file. These #EXTENSION templates generate code every time the APP is regenerated. For this template, I chose to write a #UTILITY template. A #UTILITY template is an on-demand template. Output is generated only when the template executes. I didn't see the point of constantly generating a huge XML file every time I made a simple change to an APP file. The only downside is that because #UTILITY templates are not attached to an APP file, their #PROMPT values are not saved between executions.

My template starts off like this:

```
#UTILITY(DictionaryToXml, 'Write Dictionary to XML')
#PROMPT('Output File', SAVEDIALOG('Output File', '*.*')), %pOutputFile, REQ
#PROMPT('Dictionary Info', CHECK), %pExportDictionaryInfo, DEFAULT(%True)
#PROMPT('File Info', CHECK), %pExportFileInfo, DEFAULT(%True)
#PROMPT('Key Info', CHECK), %pExportKeyInfo, DEFAULT(%True)
#PROMPT('Field Info', CHECK), %pExportFieldInfo, DEFAULT(%True)
#PROMPT('Relation Info', CHECK), %pExportRelationInfo, DEFAULT(%True)
#PROMPT('File Driver Info', CHECK), %pFileDriverInfo
#INSERT(%Notice)
```

The only interesting bit shown here is the #INSERT command. #GROUP(%Notice) is defined elsewhere in the template file. The #GROUP contains my basic "don't sell my work" message, and I include it in all my templates. (You might also note that I default every checkbox to checked, except %pFileDriverInfo. This is done just so I don't have to check them all each time I execute. The %FileDriver symbols contain information about each file driver that is registered with the IDE. I included the symbols for completeness, but

don't find it particularly useful for my purposes.)

By the way, #GROUP(%Notice) contains the following PROMPT:

```
#PROMPT('Updated: ', @D4), %pDateModified, PROP(PROP:Readonly, 1),
     PROP(PROP:Trn, 1), PROP(PROP:Skip, 1)
```

I only point this out to demonstrate the new template language PROP syntax. #PROMPTs are used to gather information from the user. But in this case, I want to display a value and keep the user from changing it. Before the PROP syntax, a readonly PROMPT wasn't possible.

Next, I set up a couple of default values using a #PREPARE section (a #PREPARE section is executed before the template's setup screen is displayed, but after the screen's template symbols are created):

```
#PREPARE
  #SET(%pOutputFile, %DictionaryFile & '.xml')
  #SET(%pDateModified, DEFORMAT('August 1, 2005', @D4))
#ENDPREPARE
```

%pOutputFile is declared above as a #PROMPT. I'm simply defaulting the value to whatever the DCT's name is and appending the .xml extension (e.g. books.DCT => books.DCT.xml). %pDateModified is also defaulted, and you'll find it declared as a #PROMPT within the #GROUP(%Notice). This little trick (defaulting a #PROMPT that is declared elsewhere) allows me to have a different date for each template where I #INSERT(%Notice).

Now, I'll declare a couple of symbols to be used while I loop through the dictionary:

```
#DECLARE(%Groups), MULTI, UNIQUE
#EQUATE(%nFiles, ITEMS(%File))
#DECLARE(%nFile)
#SET(%nFile, 0)
```

I'll use the %Groups symbol to keep track of nested fields (for example, you might have a STRING, which is nested within a GROUP, which is nested within another GROUP, etc), and I'll explain more about it near the end of the article. The %nFiles symbol caches the total number of files in the dictionary (this is done for performance). I use the #EQUATE command, rather than the #DECLARE command, because the value of this symbol will not change. And, %nFile is used to display some progress to the user, as you'll see shortly.

Finally, I can start generating some XML:

```
#CREATE(%pOutputFile)
<?xml version="1.0"?>
<Dictionary>
  #IF(%pExportDictionaryInfo)
  #INSERT(%GetDictionaryInfo)
  #ENDIF
  <Files>
  #FOR(%File)
```

```
#SET(%nFile, %nFile + 1)
#MESSAGE('Processing ' & %nFile & ' of ' |
   & %nFiles & ' (' & %File & ')', 1)
<File>
```

I'll walk you through each line of code. The #CREATE command creates the file (remember, the %pOutputFile was defaulted in the #PREPARE section, but the user may have changed it). If the file already exists, I just replace it (see the docs for #OPEN if you'd rather append). Also, the #CREATE command puts the new file "in scope." Everything generated from now on is generated in this file. To stop generating to this file, you can either #CLOSE the file, or you can #CREATE or #OPEN a different file. #CREATEing and #OPENing files is a stack-based operation. For example, if you #CREATE or #OPEN a second file, it becomes "in scope," and pushes itself onto the "files in scope" stack. When the second file is #CLOSEd, the first file is popped back to the top of the stack.

After #CREATEing the file, the first line generates the required XML declaration. The second line generates the opening <Dictionary> tag. If the user checked the %pExportDictionaryInfo PROMPT, the #GROUP(%GetDictionaryInfo) is called. I'm going to skip over the details of the #GROUP(%GetDictionaryInfo) for now. An opening <Files> tag is generated, and a #FOR loop is started to loop through all of the %File symbols.

The %nFile symbol is incremented, and a #MESSAGE is used to update the user of the template's progress. This #MESSAGE command generates something that looks like this: "Processing 5 of 103 (Authors)." The second parameter tells the Clarion generator to place this message on the first line within the MessageBox (there are three lines available to a template author).

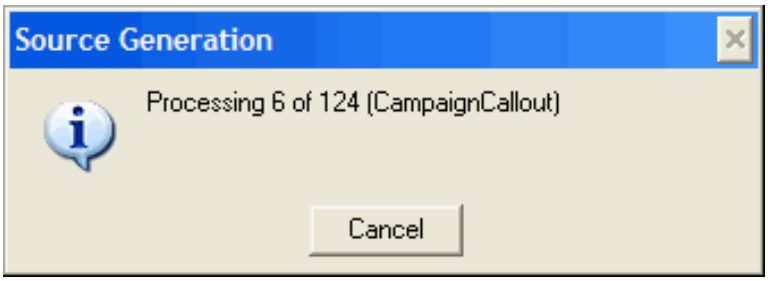

Figure 2. Displaying source generation progress

And finally, an opening <File> tag is generated.

The next line is fancy:

```
%(%WriteTag('Label', %GetStringValue(%File)))
```

So, what's going on here? Let's review the different methods for invoking a #GROUP template:

1. #CALL(symbol[(set)][, parameters])[, returnvalue]
2. #INSERT(symbol[(set)][, parameters])[, returnvalue][, NOINDENT]
3. #INVOKE(symbol[, parameters])[, returnvalue][, NOINDENT]
4. CALL(symbol[, parameters])

5. INVOKE(symbol[, parameters])
6. %(expression)

Here's my quick explanation (see the docs for a more detailed explanation). The #CALL command calls the #GROUP named by symbol[set], passes the parameters (if necessary), and populates the return value (if you supplied a symbol to hold it). #INSERT does the same as #CALL, but provides the NOINDENT option. #INVOKE does the same as #INSERT, but requires 'symbol' to be a variable.

CALL can be used within another template expression, such as:

```
#SET(%myvar, CALL(%MyGroup))
```

INVOKE is the same as CALL, but allows 'symbol' to be a variable, such as:

```
#SET(%aGroup, '%MyGroup')
#SET(%myvar, INVOKE(%aGroup))
```

The last option is my favorite, because it looks the most like real code. It allows you to call the #GROUP inline, and that's what I'm doing here (getting back to the subject):

```
%(%WriteTag('Label', %GetStringValue(%File)))
```

Okay, here's where my template gets really slick, so, pay attention. Obviously, I'm calling the %WriteTag #GROUP:

```
#GROUP(%WriteTag, %TagName, %TheValue)
  #RETURN('<<' & %TagName & '>' & %TheValue
    & '<</' & %TagName & '>')
```

This #GROUP simply generates something like this (you'll notice that I've doubled-up the left-angle bracket; this is per Clarion syntax convention):

```
<TagName>TheValue</TagName>
```

So, the %TagName parameter is Label, but to get the %TheValue parameter, I've inlined another #GROUP:

```
#GROUP(%GetStringValue, %TheValue), AUTO
  #DECLARE(%sValue)
  #SET(%sValue, CLIP(LEFT(%TheValue)))
  #CALL(%ReplaceSubString, %sValue, '&', '&amp;')
  #CALL(%ReplaceSubString, %sValue, '<<', '&lt;')
  #CALL(%ReplaceSubString, %sValue, '>', '&gt;')
  #RETURN(%sValue)
```

Basically, this function is making the string XML-friendly. First, LEFT and CLIP are used to remove all leading and trailing spaces. The %ReplaceSubString #GROUP is called to replace '&', '<', and '>'. These three characters are special, and must be replaced with the XML 'entities', as shown in the #CALL statements.

If you don't replace these special characters, every XML parser on the planet will get confused and puke all over your XML. So, let's look at %ReplaceSubString:

```
#GROUP(%ReplaceSubString, *%pString, %OldSubString,
    %NewSubString, %pCaseSensitive = 0), AUTO
  #DECLARE(%lPosition, LONG)
  #DECLARE(%lLen, LONG)
  #SET(%lPosition, 1)
  #SET(%lLen, LEN(%OldSubString))
  #LOOP
    #IF(%pCaseSensitive)
      #SET(%lPosition, INSTRING(%OldSubString,
        %pString, 1, %lPosition))
    #ELSE
      #SET(%lPosition, INSTRING(UPPER(%OldSubString),
         UPPER(%pString), 1, %lPosition))
    #ENDIF
    #IF(%lPosition = 0)
      #BREAK
    #ELSE
      #SET(%pString, SUB(%pString, 1, %lPosition - 1)
        & %NewSubString & SUB(%pString, %lPosition + %lLen,
        LEN(CLIP(%pString)) - %lPosition))
      #SET(%lPosition, %lPosition + LEN(%NewSubString))
    #ENDIF
  #ENDLOOP
```

The only thing fancy here is the first parameter (*%pString). Yes, you can pass parameters by reference in templates. There is no need to copy the string, and return the copy. Just modify the string that was passed in. When the #GROUP is done, the modifications will still be in affect.

So, we're finally back to generating the tag:

```
%(%WriteTag('Label', %GetStringValue(%File)))
```

%GetStringValue used %ReplaceSubString to make the value of %File XML-friendly, and %WriteTag generated a proper XML tag. Very nice.

We're almost done now. Let's go back and look at the details of the #GROUP(%GetDictionaryInfo):

```
#GROUP(%GetDictionaryInfo)
%(%WriteTag('FileName', %GetStringValue(%DictionaryFile)))
%(%WriteTag('QuickOptions', %GetQuickOptions(%DictionaryQuickOptions)))
%(%WriteTag('UserOptions', %GetUserOptions(%DictionaryUserOptions)))
%(%WriteTag('ToolOptions', %GetToolOptions(%DictionaryToolOptions)))
%(%WriteTag('Description', %GetStringValue(%DictionaryDescription)))
%(%WriteTag('Version', %GetVersionInfo(%DictionaryVersion)))
%(%WriteTag('TimeCreated', %GetTimeValue(%DictionaryTimeCreated)))
%(%WriteTag('DateCreated', %GetDateValue(%DictionaryDateCreated)))
%(%WriteTag('TimeChanged', %GetTimeValue(%DictionaryTimeChanged)))
```

```
%(%WriteTag('DateChanged', %GetDateValue(%DictionaryDateChanged)))
```

As you can see, this code uses %WriteTag to do all the heavy lifting. Other functions make sure the values are XML-friendly (%GetStringValue, %GetTimeValue, %GetDateValue, and %GetBooleanValue). (%GetQuickOptions, %GetUserOptions, %GetToolOptions, and %GetVersionInfo are currently just stub functions. If some kind soul would be interested in discovering and parsing each particular syntax, I would be appreciative. I just haven't had the time and the need yet to do these myself.)

I won't bother explaining the rest of the #GROUPs in the template, because each is written in a manner consistent with what I've already covered. If you're interested, just look at the template itself. Each checkbox PROMPT correlates to a #GROUP:

```
%pExportDictionaryInfo --> %GetDictionaryInfo
%pExportFileInfo       --> %GetFileInfo
%pExportKeyInfo        --> %GetKeyInfo
%pExportFieldInfo      --> %GetFieldInfo
%pExportRelationInfo   --> %GetRelationInfo
%pFileDriverInfo       --> %GetFileDriverInfo
```

The only other interesting part of the template occurs while looping through the %Fields within a %File. If you recall, I defined a multi-valued symbol near the top of the template:

```
#DECLARE(%Groups), MULTI, UNIQUE
```

This line basically states that %Groups is a symbol that can store multiple values, and each value is unique. The #FOR, #FIX, #FIND, #ADD, #DELETE, #SELECT, and #FREE commands can all be used with a multi-valued symbol. Likewise, the INLIST, INSTANCE, and ITEMS functions are available (see the docs for more information).

The template uses the %Groups symbol to keep track of nested fields:

```
#FOR(%Field)
  #IF(NOT %FieldID)
    #POP(%Groups)
  #ELSE
    <Field>
    #INSERT(%GetFieldInfo)
    </Field>
    #IF(%FieldType = 'GROUP')
      #ADD(%Groups, %Field)
    #ENDIF
  #ENDIF
#ENDFOR
```

If %FieldType = 'GROUP', the template #ADDs a value to %Groups. When 'NOT %FieldID' is true, meaning %FieldID is blank, the template knows that it's reached the end of a GROUP, so it #POPs the current value from %Groups. The #POP command automatically selects the new last item in the multi-valued symbol.

So much for the dictionary. Next week I'll look at how to export the APP file.

Download the source

Harley Jones lives in Mobile, AL. He's been programming in one way or another since he was ten, and seems to have a knack for it. After graduating college with a degree in English Literature, he was hired as a programmer, and was introduced to Clarion. Currently, he's buried in .Net projects where he's constantly looking for methods to streamline production with code generators.

Reader Comments

Add a comment

- » That's how the TXD and TXA should be! The next logical...
- » Good ! As Clarion Mag is accessed all over the world,...
- » Excellent question! As far as I know, the Clarion IDE...
- » Using the template of yours (with default encoding to...
- » Hey Cleverson, Wow. Discovering that the Clarion IDE...
- » Harley has supplied a newer version of the template based...

Clarion Magazine

Clarion News

Search the news archive

SoftVelocity enters the blogosphere

SoftVelocity has a blog now, and right now it contains two entries by Bob Zaunere. One entry is a brief note on the 9048 patch, and the other is a lengthier comment on the features in IP Driver 2.0 (not yet released, but coming later this week).
Posted Thursday, October 20, 2005

C6 9048

Information on the 9048 patch is going out by email, but the download page has not yet been updated.
Posted Thursday, October 20, 2005

TIFF-Change 3.23

Version 3.23 of TIFF-Change for Clarion with revised Templates and Clarion Demo apps is now available. TIFF-XChange allows Clarion developers to generate Mono/Grayscale/Color compressed TIFF image files from Clarion Reports and other 3rd party applications (such as MS Word/Excel/AutoCAD etc) controlled via their application.
Posted Thursday, October 20, 2005

Clarion Developer's Challenge Week #6 Winner Is Ian Cook

The Clarion Developer's Challenge Week #6 Winner Is Ian Cook. Ian picked an incredible 13 of a possible 14 winners, and is the winner of Peter Rakké's of RADventure RADIntelliSense. 2nd Place with 11 correct picks and a little from the tie breaker is Larry

Juker. Larry is the winner of Charles Edmonds' of LANSRAD EZRound.
Posted Thursday, October 20, 2005

IP Driver Web Hosting

Clarion Webhosting is offering two new IP Driver web hosting packages for the new IP Driver version 2. Package #1: Unlimited bandwidth, 100Mbit feed, 1Gb webspace. Daily Backups. Package #2: Unlimited bandwidth, 100Mbit feed, 1.5Gb webspace, Hourly Backups, Mirror Server Account for redundancy. Free Email & Skype Support. Free C6 ABC Feature & Bug Tracking Addon to add to Clarion apps. Signup before the release day of IP Driver version 2 and you get: Free account setup usually £25; 10% discount off the monthly rate for the first year. Your 12 month contract will only start from the release day so you get from now until the release day as free server time. If you would like an IP Driver web hosting account or have any further questions please email Richard@isv1.com.
Posted Tuesday, October 18, 2005

PrintWindow 1.00 Beta Build 94

PrintWindow takes the information from the controls of your window, and with that information creates them dynamically on a report. This way you have a lot of control over what is actually printed. Just add a global extension to your program, and when you run your program you can press a hot key on any window and obtain a Printout of it (new on build 88). Each Tab in a separate page (if you want), graphic background support to emulate filling forms, and much more. If you don't want to print all the windows but a few ones, you can use a different global extension, to allow you to manually populate a local control template with a button to print the window (instead of a hot key). Now with many improvements in the listbox area. Basic tree support, foreground and background colors, and styles for cells (including typeface, style, color and size).
Posted Tuesday, October 18, 2005

FullRecord 1.02

Changes in FullRecord 1.02: Fixed missing primary file on ABC English version of the template on RecoverAudit control template and on ConsultaAudit control template.
Posted Tuesday, October 18, 2005

NeatMessage 1.02

New in NeatMessage 1.02: Vertical offset (to "clear" an area on the upper part of the window); Prompts for buttons text translation (for internationalization).
Posted Tuesday, October 18, 2005

MS Visual Studio Launch and Free Product

A reminder from Arnor Baldvinsson about the upcoming VS.NET product launch tour. In the US, participants can get a free copy of the Professional version of Visual Studio along with the new MS SQL server 2005.
Posted Tuesday, October 18, 2005

CIA Factbook Download

Roberto Artigas has updated the download for the CIA Factbook. The download now includes a test application, dictionary and executable for anyone who wants to play with the data. For those of you that registered and picked up just the data, drop by again and pick up the whole package. You will find it a bit much easier to deal with the data. The TXD that the web scrape program created is also include so you can add it to your own applications.
Posted Tuesday, October 11, 2005

xToolTipPro 1.0

This class and extension template allow you easily change standard tips into native Microsoft tooltip controls with many additional features. Main features: You can set and change in runtime, background and foreground color for tooltips, separately for global and local tooltips; You can have different tooltips on the window; You can make fixed tooltips. And show it automatically over any control of window or in any position of the screen; You can set left, top, right and bottom margins for tooltip text; There are many code templates for ease of use. Support SingleExe, MultiDll (Local Mode, Standalone Mode), 32-bit. Compatible with Clarion 6.2 (build 9047), Clarion 6.1 (build 9034) Compatibility with Clarion 5.5 is possible, but is not guaranteed 100%. xToolTipPro 1.0 already available on ClarionShop for $69.
Posted Tuesday, October 11, 2005

Clarion Developer's Challenge Week #5 Winner

The week #5 Clarion Developer's Challenge winner of XP-TaskPanel ($249) from Gøran Thomassen of PowerOffice is Dean Burgess. Congratulations to Dean and thanks to Gøran

for being the vendor of the week.
Posted Tuesday, October 11, 2005

DevDawn Blog

DevDawn is a site which is run by techos who just happen to be Clarion Developers. Their mission for this site is to give a broader picture to what developers do in software development.
Posted Tuesday, October 11, 2005

FinalStep 2.0 Split Up - NeatMessage Emerges

FinalStep has become too complex as a single product, so it is being divided up into subproducts. The first one is a message box replacement called NeatMessage. This new product replaces the message replacement present in FinalStep 1.5 and adds newcharacteristics and better documentation. This is free for current owners of FinalStep and a special low price to new users this month. Some features are pending in release 1.00, but still as it is, it's better than the one included in FinalStep 1.5.
Posted Tuesday, October 11, 2005

TipLink Tool Suite

LANSRAD has released the TipLink Tool Suite, a combination of TipLink Reporter and TipLink TipSynch. These tools let you export and analyze all the tooltip data from your application using an XML file and an analysis program that runs after the export is completed. Price is $24.95 USD.
Posted Tuesday, October 11, 2005

UK Clarion User Group Meeting Nov 21

The next UK Clarion User Group meeting is on Monday Nov 21st at Kings Langley. See the web site for details including travel and accomodation info. New Members are always welcome, and the user group now has several overseas members as well. Book now as over as two thirds of the places were filled on the first day.
Posted Friday, October 07, 2005

Whitemarsh Metabase 6.8

The Metabase Software System's latest version, v6.8, includes big improvements such as

an expanded client server layer of human-work saving processes, complete metabase data storage through a SQL engine, and a true multi-user environment for entry, update, and reporting. A item of great interest for Whitemarsh is the start of a Data Management Maturity (DM3) Assessment and Prescription process. The DM3 Assessment data model is complete and the database application that will hold, analyze, and report assessment results has begun. Recently posted to the Website's SQL section is the October 2005 SQL 200n current base documents. Also posted to the web is the SQL/XML document for the Final-Draft International Standard. SQL/XLM will likely be a final standard by Spring 2006.
Posted Friday, October 07, 2005

Three Way Tie In Week Four Of Clarion Developer's Challenge

Week #4 of the Clarion Developer's Challenge has three players tied at ten correct picks: Dean Burgess, Jerry Davis, ands Ed Schneider. Congratulations to this weeks winners and a big thank you to the Clarion Developer's Challenge vendors of the week, Comsoft7, BoxSoft, and SoftVelocity.
Posted Friday, October 07, 2005

xCheckTPS 1.04

xCheckTPS is a free utility for checking TPS files for errors. Features include: You can check both single file and group of files; You have log of checking; Program have some settings for more comfortable using; Program have runtime command line and keys. Works with non-encrypted TPS files with no password. xCheckTPS is based on TPSFile class by Vladimir Yakimchenko.
Posted Friday, October 07, 2005

SysTree C6.2 Compatible

A Clarion 6.2 compatible version of SysTree is now available. Version 1.3.2 contains files for all Clarion 6 versions (6.0, 6.1 AND 6.2) plus the old files for 5.0 and 5.5. This is not a feature update. Registered users can download the update free of charge. The installation password has not changed, so please use the data sent to you upon purchase.
Posted Friday, October 07, 2005

BSPrintList 1.0

BSPrintList was developed primarily as a way to generate WYSIWYG reports for the BST

template listboxes.
Posted Friday, October 07, 2005

dpQuery 2.06

dpQuery 2.06 has been released. This version fixes missing mapped fields in saved queries.
Posted Friday, October 07, 2005

Up & Up 1.2

Version 1.2 of Up & Up adds support for recursive (self-related) tables to Up & Up. This feature requires UltraTree version 8, and the UltraTree Recursive Views feature, in addition to Up & Up. The feature has been packaged as a separate add-on to allow Up & Up to be used without this feature by those who don't have UltraTree or the required UltraTree feature. With this capability, Up & Up now provides complete processing and reporting of all file structures supported by UltraTree. Both normal and rollup totaling is fully supported for recursive tables. Normal totaling totals across the records of a single recursive level. Rollup totals accumulate across all levels. Rollups can be accumulated for any particular level using conditional totaling.
Posted Friday, October 07, 2005

iQ-XML 1.10

There is a small update to iQ-XML now available. This version will now handle files picked up from Unix/Linux/MAC/Mainframe files. When parsing, the system will automatically detect EndOfRecord indicators with ASCII 13/10, ASCII 10 or ASCII 13. iQ-XML is a free tool for Clarion developers to add XML functionality to their applications with very little knowledge. iQ-XML comes with both Parser and Writer functions. Quickly generate an XML document from a Clarion Queue, Group structure, or just using the API's. A novice user can also read a complex XML document and fill a Clarion Queue. Navigate easily through-out the document, finding nodes and parsing only what you need. For users who feel comfortable with Templates, there are also templates options available for all the API's. Both PDF, Online HTML documentation, as well as example applications are also included.
Posted Friday, October 07, 2005

solidsoftware Atom Feed

An Atom (1.0) feed with news of solidsoftware product releases is now available.
Posted Monday, October 03, 2005

solidsoftware RSS Feed

An RSS (2.0) feed with news of solidsoftware product releases is now available.
Posted Monday, October 03, 2005

SysList Clarion 6.2 Compatible

A Clarion 6.2 compatible version of SysList is now available. The new version 1.3.3 contains files for all Clarion 6 versions (6.0, 6.1 AND 6.2) plus the old files for 5.0 and 5.5. This is not a feature update. Registered users can download the update free of charge. The installation password has not changed, so please use the data sent to you upon purchase.
Posted Monday, October 03, 2005

xTipHotKey 2.9

New in xTipHotKey 2.9: Modifications in template for compatibility with Data Conversion Templates from SoftCreator.
Posted Monday, October 03, 2005

Oz DevCon Survey

It has been a while since Australian developers have got together for a DevCon type meeting. It used to be called ConVic when it was held in Victoria. The Perth Clarion User group is keen to see a 2006 conference take place, and are even prepared to host it in Victoria. To help the user group assess the viability of such an event, please complete the survey at a special web site. If anyone from overseas is interested in coming then please feel free to express your desire for doing so. Presenters are also wanted.
Posted Monday, October 03, 2005

Data Down Under Distributes SetupBuilder

Data Down Under of Eden, Australia is now the distributor for the SetupBuilder product line throughout Australia and New Zealand.
Posted Monday, October 03, 2005

Clarion Developer's Challenge Week #3 Winners

With 9 correct picks, Bob Foreman Is The Clarion Developer's Challenge Week #3 Winner! Bob, a winner last week, has donated the Softvelocity In Memory Driver prize back into the CDC prize pool and it will be offered as a week #4 prize. The second place winner is Dave Bratovich. the tie breaker was needed to break a 6 way tie! (39). Dave is the winner of Bill Roe's Valutilities!
Posted Monday, October 03, 2005

Free CWPlus Wallpapers

Ingasoftplus has made 14 free CWPlus wallpapers available to download. Sizes are 1024 x 768 and 1600 x 1200 and can be freely used. Kindly designed by Mixer.
Posted Monday, October 03, 2005

Clarion Magazine

The ClarionMag Blog

Blog Categories

- »All Blog Entries
- »Clarion 7 Clarion.NET
- »Future Articles
- »News flashes
- »Nifty Stuff

Clarion and Beyond Compare

Direct link

Posted Monday, October 03, 2005 by Dave Harms

Phil Carroll at Enabling Simplicity (think UltraTree) has created some file comparison rules for those of us using Beyond Compare. Among other things these rules tell BC's compare utility to ignore case and whitespace differences, as well as comment lines (although as comments are defined as anything following the ! character, I suppose there's a slim chance a change with a ! string literal could slip through). Phil sez these rules may be included in release 2.4. Meanwhile you can download them yourself.

I completely agree with Phil on the usefulness of Beyond Compare. Among other things, I use BC to back up my system to removable hard drives, using BC's batch mode.

What the heck is Web 2.0?

Direct link

Posted Tuesday, October 04, 2005 by Dave Harms

Bill Kinnon points at this article by Tim O'Reilly titled What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. Web 2.0 is one of those nebulous concepts that seems like it has the potential to completely reshape software development as we know it. Software is increasingly living on the web, and interestingly, databases are a massively important part of that movement.

All of this, I think, only makes Clarion.NET that much more important. Win32 just doesn't offer the same flexibility for web-related development. And clearly Clarion developers have already got the message that the web is massively important to their futures, as indicated by a couple of recent ClarionMag polls.

I'm mulling over an article on Web 2.0 and Clarion...

A Spring DevCon?

Direct link

Posted Wednesday, October 05, 2005 by Dave Harms

There's a rumor going around that the next Clarion DevCon will be in March or April. Time to crank up the mill and see what this would mean, if true.

If we do get an announcement shortly, that will tell me that C7 and Clarion.NET both must be close to beta (whether or not they are actually released as beta). Here's why. Z's been burned enough on due date announcements that I can't see him risking a conference without having product pretty much in hand, but a full DevCon needs at least six months

lead time to get facilities, speakers, and attendees lined up. Even a smaller tech briefing ought to have a three month runup. You can't just say "Okay, we finally have some good stuff here, let's have a DevCon next month," and neither is it wise to risk announcing an event even six months down the road when there are showstoppers still to be resolved, with no clear (and short) path to resolution.

To me, a DevCon announcement is a clear signal that the new code is rounding into shape. And personally, I don't think a conference just for C7 is adequate - there has to be a significant Clarion.NET product there as well.

On another note, if DevCon does happen in the spring, I'd vote to make the new slot permanent; with the hurricane cycle in its strong phase, and the recent catastrophic events in the gulf, it's time to move out of the cheap season.

Yo, 3P People, Read This!

Direct link

Posted Friday, October 07, 2005 by Dave Harms

Here's a word for any third party developers who want to get their product announcements up in the ClarionMag news section in the shortest possible time.

First, only a few of you actually email me news items. Mostly I harvest the SV newsgroups, and when I get a free moment I boil them down into a newsy format. This rendering is what takes the time (not that I'm suggesting your news items are dead animals turned into lard - at least not usually). So I'll make you a deal. Send me news items that meet the following requirements, and I'll post them as soon as I get them.

Format - three paragraphs (the first two are single lines), in plain text, as follows:

1. **Title**

   Product version numbers should not include "v" or "Ver" or "Version", just the product number, as in MyGreatProduct 2.1.3.

Do not abbreviate the product name

Use title capitalization (e.g. My New Product Is Now Available).

2. **URL**

   One and only one URL. If you need to present more than one URL, then you need two news items.

   The complete URL, including the http:// (or whatever protocol)

   Use the product URL please, *not* the purchase or download URL.

3. **Description**

   One and only one paragraph, max 250 words.

   Use full the product name at least once.

   Be factual - say what the product does, not how great it is. Do *not* use exclamation marks.

   Plain text only - no formatting or URLs in the body.

   In (North) American English, most nouns are *not* capitalized. Words like template and class, for example, are lower case. Only capitalize these if they are part of the actual product name (e.g. Super Templates).

   Do not use ALL CAPS under *any* circumstances..

Review the current news items for examples of acceptable news content.

If you want your announcement to show up promptly, follow these guidelines, and email me the item. I will continue to harvest news items and massage them into the ClarionMag format. But that takes time, and occasionally I come across announcements that are so

incomplete or incomprehensible that I simply don't bother.

Experimental RSS Feeds

Direct link

Posted Friday, October 07, 2005 by Dave Harms

I have two new RSS feeds up on the site: a feed for this blog, and another feed for all ClarionMag items. Kick 'em around and let me know how they work. The blog feed was up briefly with full text but no CDATA section so if you got some really ugly HTML you'll need to refresh or recreate the feed in your RSS reader.

For the basics of RSS see this article.

Canuck Turkey Day

Direct link

Posted Friday, October 07, 2005 by Dave Harms

Up here in the Great White North harvest is already finished. Among other things, that means we celebrate Thanksgiving a month or so before our American cousins. The Clarion Magazine office will be closed on Monday, October 10, 2005. And with three turkey dinners scheduled for the weekend (we have a *lot* to be thankful for), I'll probably be cataleptic from tryptophan poisoning for most of next week. Which is okay - I can use the sleep.

Happy Thanksgiving to all our Canadian friends and Canuck Turkey Day sympathizers!

Google Map your tracert

Direct link

Posted Tuesday, October 11, 2005 by Dave Harms

Mark Riffey points out this Italian page that lets you create a Google map of a traceroute. In a DOS window, run tracert (e.g. tracert www.clarionmag.com) and the highlight the resulting trace with the mouse and right-click to copy to the clipboard. Paste into the text box and click the Start button.

The map is pretty small, and it's a world map, so it won't tell you much about traffic around your own city. But if you want the detailed listing of the stops along the way you can look at the page source for the script that plots the map points.

The only problem I found is that the server chokes on Class C private addresses (192.168.0.0 - 192.168.255.255) and displays a number format exception. Not very friendly. So remove those lines, if present, from the head and tail of the trace.

The site name is interesting - wereporters.com. Given my knowledge of Italian, it could be that they're news guys, or maybe just lycanthropes who work for Trenitalia.

Also from Mark - this page on using the Google Maps API.

Rate your apps

Direct link

Posted Tuesday, October 11, 2005 by Dave Harms

Ron Childs posted a link in the SoftVelocity chat newsgroup to an online rating calculator which you can use to evaluate your software according to the Tucows Rating Guide. As Ron says, "Prolly a very good check list for any kind of software."

October 2005 articles

Direct link

Posted Tuesday, October 11, 2005 by Dave Harms

Some of the articles slated for October are:

- Maarten Veenstra demonstrates techniques for dealing with massive numbers of controls on a single window
- Harley Jones looks at creating XML from Clarion dictionaries and apps
- Ozzie Paez on the role of Clarion in the War on Terror - a very cool look at how useful Clarion is when you have a lot of data structures that are evolving rapidly.
- Phil Will on internationalization standards for third party developers
- Geoff Bomford has some cool template and API tricks.

Feel better now, Ben?<bg>

Blog Bugfix

Direct link

Posted Thursday, October 13, 2005 by Dave Harms

I've updated the blog code to fix a bug with blog categories not displaying correctly. Now when you choose one of the categories you will only get entries for that category.

Wikipedia - so comprehensive, it includes a Clarion entry

Direct link

Posted Friday, October 14, 2005 by Dave Harms

You've probably heard of wikis by now - these are user-editable web pages, and I have to say that when I first heard of the idea I was a bit bewildered. How the heck can you rely on a document that's like a giant chalkboard, where anyone can come along and add new, unverified material, or modify what anyone else has written? And more importantly, how do you get people to contribute?

As it turns out, wikis can be enormously useful, and are sometimes extremely well supported by the public. Just take a look at Wikipedia, the free, user-maintained encyclopedia. Google has long been my mainstay for web research, and more and more I'm finding Google leads me to Wikipedia.

Wikipedia even has an entry for the Clarion language. And you can easily add your own Clarion-related link, as I've just done for Clarion Magazine.

IP mapping and web tricks

Direct link

Posted Monday, October 17, 2005 by Dave Harms

Wolfgang Orth sent me a link to another site that puts IP addresses on a world map. But as Wolfgang points out, you can't always trust what you see on the Internet. ClarionMag is in Australia? The map does correctly show the location of EV1, where ClarionMag is hosted. The location marker is a wee blue cross, so you'll have to look closely.

In the past, when I needed to do some domain digging, the first place I looked was SamSpade.org. There used to be a lot of handy tools here, but sadly many of them are now broken. One I really liked was the safe browser - if you had a suspect URL, you could plug it in here and see the resulting page as source. SamSpade's safe browser has gone south, but happily there's Web-Sniffer, which used to be available through Mozilla.org but now has its own domain. Web-Sniffer also supports Internationalized Domain Names (IDNs). I'd never even heard of IDNs before, but of course it makes complete sense to be able to write domain names in any language, just as you can write web pages in any language. IDNs still have the usual top level domain ending, such as .com or .net.

.NET LINQ - the future of database queries?

Direct link

Posted Tuesday, October 18, 2005 by Dave Harms

A lot of Clarion developers still seem to be wondering why they should care about .NET, and Clarion.NET. For some of my reasons, see my article .NET Basics: What Is .NET, And Why Should I Care? Then take a half hour or so to watch the video of Anders Hejlsberg talking about, and demonstrating, Microsoft's new LINQ technology. Hejlsberg is the guy who wrote Turbo Pascal, and was the brains behind Delphi before he jumped ship to MS to head up J++, the Windows Foundation Classes, and C#.

LINQ stands for Language INtegrated Query, and is basically a way of lifting database (and other) queries up to the same level as the programming language(s) from which the queries are called. For instance, right now if you want to use an SQL statement in your app, you typically include that statement in quotes, because it's a string that will be passed to the back end. There's no compile time checking of the SQL, and the IDE knows nothing about the statement's syntax. As Hejlsberg points out, there there's a huge disconnect between programming languages and query languages like SQL.

LINQ attempts to bridge that disconnect by providing a standardized .NET query grammar (currently the examples are C#, but LINQ will be part of the .NET framework itself). You write the query statement, and the .NET runtime builds the necessary classes behind the scenes to carry out the query and return a result. This is very cool because:

- The query syntax is language-level, meaning the IDE can help you write the query (using statement completion), and find problems before the query is executed
- You can use LINQ with a variety of data sources, including relational databases, XML documents, collections (think queues of objects)
- You can mix and match these data sources
- LINQ works with classes that represent data, so you can even use LINQ to query any classes or collections of classes

The database-specific API is DLinq, the XML API is XLinq. XLinq in particular gives you the capabilities of XPath and XQuery.

Hejlsberg demonstrates some very slick transformations of data into XML, and also shows how DLinq can model an entire database as a hierarchy of objects.

I can imagine a future version of Clarion.NET that uses LINQ rather than database drivers or ADO.NET. Powerful stuff.

The future of web development

Direct link

Posted Thursday, October 20, 2005 by Dave Harms

The web is shaping up to be the next big battleground for programming languages, particularly on the client side, as developers rush to create rich browser-based interfaces. And there are at least three strong contenders.

AJAX is getting a lot of press these days. This is a combination of JavaScript and XML, and what I find interesting about that is JavaScript is actually Netscape technology. It was originally called LiveScript, and was renamed to JavaScript in a marketing deal with Sun. The old saying is that the only thing JavaScript and Java have in common is the first four letters.

PHP continues to grow in popularity. Personally I've never been a fan of PHP - I'm more of a Java snob. But client-side Java, in the form of applets, never took off, and I've only used Java for server side work. That may be changing, however, as Google seems to be getting behind Java for client side work.

And in addition to Java, PHP, and AJAX there's Macromedia's Flash, but as a proprietary technology it's a bit of a dark horse, I think.

Not always mentioned in these discussions is the back end server software, which can be just about anything as long as it's capable of delivering the client side code.

SoftVelocity enters the blogosphere

Direct link

Posted Thursday, October 20, 2005 by Dave Harms

SoftVelocity has a blog now, and right now it contains two entries by Bob Zaunere. One entry is a brief note on the 9048 patch, and the other is a lengthier comment on the features in IP Driver 2.0 (not yet released, but coming later this week).

RSS feeds are available, by category. There's no main feed listed, so use this link to get all articles (this URL is the same as the category feeds but without the GroupID).

DevCon Down Under? Survey says...

Direct link

Posted Friday, October 21, 2005 by Dave Harms

Tony York and the Perth Clarion User's Group are spearheading an effort to bring a Clarion DevCon to Australia in 2006. Bruce Johnson, Russ Eggen, and Gus Creces have all indicated their willingness to appear as international presenters. And of course there's a lot of local talent. I was privileged to attend, and speak at, the 1998 conference in Ballarat, near Melbourne, and there have been a number of conferences held in the state of Victoria since that time. ClarionMag has reports on ConVic '99 and ConVic 2003. Both of these are free articles.

Tony needs some help, however. The two locations being considered right now are Perth and Geelong. Perth is obviously more convenient for the Perth User's Group, but a Victoria location is definitely a possibility, if that's what prospective attendees prefer. As well, Chris Livingstone, organizer of past ConVics, will allow the use of the ConVic name if the conference is held in that state. Other location suggestions are also welcome.

To help sort out the best location, Tony is asking anyone interested in attending to fill out an online survey. There are two very short pages, and it will only take a minute.

For my fellow non-Australians, if you've never been to Oz, I highly recommend the trip. I was there for a month in '98, and was treated royally. Yes, it's a long flight from the northern hemisphere in particular, but you won't be disappointed. The Oz Clarionites are a great bunch, and there is a ton to see and do.

SV Blog Bizzie

Direct link

Posted Friday, October 21, 2005 by Dave Harms

Three posts so far today from the SoftVelocity blog, including a previously unpublished Quick Take from Bob Foreman on using CALL and an announcement about upcoming file driver updates. Although I'm still waiting impatiently for some news of Clarion.NET, it is nice to see a spate of activity at the SV web site (which has also been tidied up some). Keep it coming.

By far the easiest way to get the SV blog entries, and mine, is via an RSS reader. You point your reader at an appropriate RSS URL, and tell it to periodically poll the site for new items. This way you're automatically notified of new content. I'm using the Sage extension with Firefox.

Some feeds you may want to track:

- This blog
- All items from Clarion Magazine (news, articles, blogs, etc)
- All items from the SV blog (the SV blog page lists feeds for the individual topics, but is missing the overall feed link)

SoftVelocity in hurricane's path

Direct link

Posted Monday, October 24, 2005 by Dave Harms

Hurricane Wilma is crossing southern Florida, and was a category 3 hurricane just before landfall. SoftVelocity's office is in Pompano Beach which is right in the path of the storm. Although this is a fast moving storm and will be offshore by the afternoon, the area does appear to be taking some significant damage. There hasn't been any word from folks at SV yet, but it could well be a few days before the office opens again even if their own building is unaffected.

SoftVelocity changes name to HighVelocity

Direct link

Posted Monday, October 24, 2005 by Dave Harms

Winds in Pompano Beach, home of SoftVelocity, have gone as high as 120 mph this morning. Yikes! Looks like the eye is passing over and the backside is yet to come. The problem with this situation, according to veteran storm watchers, is that the winds will be going the opposite direction and all the debris that blew off the first time around can now be picked up again, and areas that were sheltered from the first half of the hurricane are no longer sheltered.

About half a million people in Broward Country (which includes Pompano Beach) are without power.

Why Clarion?

Direct link

Posted Monday, October 24, 2005 by Dave Harms

If you've used Clarion for any length of time, you're well aware of its core strength: rapidly developing, and rapidly modifying, database-centric applications. But if you've ever had to pitch a Clarion solution to someone with no knowledge of Clarion, you know it can be a tough sell.

Here's one more success story you can use. I've made Ozzie Paez's article Clarion and the War on Terror free access, and you're welcome to link to it, or make print copies solely for purposes of illustrating Clarion's capabilities (please include the link to the original article on all print editions).

I've also made the first installment of Wade Hatler's series on Clarion and .NET free access.

SV newsgroups are down

Direct link

Posted Monday, October 24, 2005 by Dave Harms

SoftVelocity's newsgroups appear to be down. I had problems early this morning and got access again, but I can't get there now. The web site and the blog site are both still up. Andrew Guidroz points out that lots of Internet traffic was rerouted through Florida after Katrina hit New Orleans, and with Wilma affecting power in Florida, the Internet is starting to go a bit wonky in Louisiana.

It may be a while before SV staff get back on the job. The main thing right now is that everyone stays safe.

SV news server back online (maybe)

Direct link

Posted Monday, October 24, 2005 by Dave Harms

SoftVelocity's news server looks to be back online. News sources are showing substantial hurricane damage in S. Florida so don't be surprised if access is spotty for the next few days.

Hurricane Wilma temporarily closes SV office

Direct link

Posted Tuesday, October 25, 2005 by Dave Harms

Bob Zaunere has posted a message in the SoftVelocity newsgroups confirming that the SV office was hit hard by hurricane Wilma. Power is still off and will not be restored for at least a few days, and possible two weeks (not surprising, since about 6 *million* people are without power right now). By now you may have seen video of the massive damage to nearby Ft. Lauderdale, and it would appear winds were even higher in Pompano. Some SV staff are working in a temporary office further north on the Florida coast.

If you did not receive an email regarding the 9048 patches or the new builds of the In-Memory and Dynamic drivers, send an email to wilma at softvelocity dot com. Z says keep an eye on the new blogs for more updates, and thank you for your patience.

SV employees safe and sound

Direct link

Posted Wednesday, October 26, 2005 by Dave Harms

Bob Zaunere has reported that all SoftVelocity employees have been located, and are safe and sound. For those of you trying to reach friends and family in the area, Z also points out that while phones are still working in some hurricane-affected areas, a lot of people have come to rely on cordless phones, and these phones will not function without A/C power. Good idea to keep at least one corded phone around for emergencies.

SV office still without power, may be weeks

Direct link

Posted Thursday, October 27, 2005 by Dave Harms

This AP story says it could take until November 22 for full power to be restored in Broward County. For now, SoftVelocity employees are working at an alternate location further north. I tried calling the Pompano Beach office, and as expected there was no answer.

Z hints at DevCon 2006 and a hand coder's alpha of Clarion.NET

Direct link

Posted Thursday, October 27, 2005 by Dave Harms

Right now there's a pretty good signal to noise ratio in the new SoftVelocity community forums. Not many folks there yet, but one of them is Bob Z.

Among other things, Z has confirmed that there will be a hand-coder's early preview alpha release of Clarion.NET. This is very good news. Although the full version with AppGen will be the real deal, I think some time getting comfortable with Clarion.NET source and the .NET environment will be most useful. And let's not forget that the first releases of Clarion for Windows were for hand coders, and at least one developer I know released commercial, hand coded product with Beta 3.

Responding to a question about DevCon timing, Z says it's "roughly planned for sometime around April."

Thunderbird, RSS, and other woes of open source

Direct link

Posted Friday, October 28, 2005 by Dave Harms

First, let me say that I've been using Mozilla Thunderbird for a few months now, and for the most part I like it. It's the first mail/news/RSS reader I've used consistently since giving up on Virtual Access (which had the singular quality of being able to handle newsgroups, mail, and Compuserve fora, back when Compuserve mattered.

I've had a few gripes along the way.

- Too many windows opening - fixed with the No New Windows on DoubleClick extension.
- Lots of bugs on filtering and searching email, mostly to do with entry fields not being accessible. This seems much better in 1.5 B2.
- Inability to search newsgroup messages, except on the server, a command not supported by the SV server. This one just blows my mind - how can you not provide a local newsgroup search facility? It's got to be exactly the same code as mail search. Grab a brain, people!
- Inability to automatically download all newsgroup messages. This is another no-brainer. Why do I have to explicitly click that little button in the

lower left hand corner to go offline and get the messages? I guess since I'm not supposed to search locally, I'm not supposed to save locally either. To be fair (well, if I have to) there might conceivably be massive newsgroups out there where I would rather not grab all the messages. But the SV newsgroups aren't on usenet so I can't search them any other way but locally. Thankfully Google Desktop will index the TB news base, but it the View in Thunderbird link doesn't seem to work.

- Adding RSS feeds can be a bit wonky. In previous versions I had to click Add repeatedly, while standing on my head and whistling the Battle Hymn of the Republic before the add new feed dialog would let me enter a URL. This seems to be fixed in 1.5 B2.
- Unbewievabwe memory usage. Typically TB would consume > 600 MEGS of memory, and frequent incredible slowdowns while the hard disk thrashed. Yikes! But after installing the latest beta, I noticed that I wasn't supposed to be installing the new version on top of the existing version. So I renamed the original directory, installed again, and now TB seems better behaved, although after a few passes through the SV newsgroups, numerous mail accounts, and a big whack of RSS feeds, it's still eating some 175 megs of memory. But no more thrashing, at least so far.
- Overall slow performance. TB isn't as snappy as I'd like - I frequently have to wait a couple of seconds for the transition from the last read message in a newsgroup to the next unread message in the next group.

There are a lot of things I like about TB. The message base is straight text, so no ugly binary formats if things go pear shaped. There are a ton of shortcut keys - I can get all the way through chat in mere seconds, just by hitting the T key to mark a thread as read. And there are lots of options for viewing messages - I prefer to set the View dropdown to All, and then select View|Threads|Threads with unread from the main menu. Of course that leads me to another gripe - I have to do this for *every single newsgroup*. More ludicrosity.

Ah well. A 1.0/1.5 release is pretty early in a product's lifetime, and all things considered TB is still a pretty good tool. I've learned to live with its idiosyncracies, and I can wait for the 3.0 release, which in classical software mythology is the magic point at which basic functionality becomes fully usable, but before the feature creeps take over.*

* Okay, not *strictly* true for consumers of the Clarion product line...

Power back on in Pompano

Direct link

Posted Monday, October 31, 2005 by Dave Harms

The power is back on at the SoftVelocity office in Pompano Beach. Among other things, this means that the Problem Tracker System is back up and Clarion developers can once again post bug reports, upload examples, and track the status of submitted reports.

[Last 25 entries] [Last 50 entries] [All entries]