CenturaPro™

Visit us at www.ProPublishing.com! Hot Ideas for Centura® Developers

SAL Gets Smart

Mark Hunter and Frank Böttcher

Centura Team Developer gets a big boost to its user interface with IntelliSal, a new third-party product from the Ice Tea Group. We caught up with the author, Frank Böttcher, just after the product was officially announced (and well received!) at the London conference. Here are our questions, and his answers.

What is IntelliSal?

IntelliSal is an enhancement tool for CTD 1.1/1.5. It offers context-sensitive assistance for available members and functions as well as an overview of function parameters, plus some extra features. It's a straightforward, easy-to-use tool and fits tightly into the outliner.

Why should CTD developers care about it?

Right now, CTD developers have only the Coding Assistant, which is quite limited in its ease of use. Any developer using a second IDE, such as Visual C++, really misses functionality like IntelliSense in the CTD outline.

SQLWindows 32

Can you estimate the productivity improvement from using it?

Especially in large projects with a big underlying class framework, IntelliSal greatly improves transparency of the application, by eliminating the need to manually scroll through big applications simply to look up functions or parameters.

What motivated you to develop IntelliSal?

After exchanging ideas with Gianluca Pivato and other members of the Ice Tea Group, I created the first version of IntelliSal. The initial goal was simply to find out what was possible. After extensive testing, tons of e-mails, and constant improvements, a full-featured IntelliSal was ready.

How is it different from the Coding Assistant?

The Coding Assistant shows far too much information, is too complicated to use, and eats up quite a lot of screen space. IntelliSal fits very tightly into the outliner, is fully useable with the keyboard, and appears only when it's needed.

It offers more features than the coding assistant:

- Autocorrection of lower/uppercase typos for internal, external, and SAL functions.
- Assistance for function parameters.

Continues on page 3

August 1999

Volume 4, Number 8

PRO Publishing

Radio London

Mark Hunter

Centura came to the London conference in July prepared to deal with a demanding European audience. And they were largely successful. For example, here's some of attendee Martin Knopp's reactions to the presentations:

"Talk about the upcoming CTD 2.0, which is now in alpha stage and targeted to be released Q1/2000, focused on the new architecture of the runtime: making it threadsafe, adding COM server generation, dynamic instantiation (yes, finally we seem to get what we asked for for years now) and, last but not least, better Web application development support. The last thing was a little bit of a surprise for me as I thought Centura Web Developer would be dead. However, Centura finally discovered the potential lying in this product. Expect improved Web classes for CTD/CWD 2.0 and a great benefit of the new runtime architecture. Finally, net.db is going to be much more integrated with CTD to integrate business logic without losing the easy-to-use approach for data publishing. The presentation would not have been as interesting if there had not been a demo of the CTD 2.0 alpha-version which showed that many of the things talked about (specially the COM server generation) already do work. For me that was the first proof that the new development process is starting to work and I really believe that 2.0 will be out on time and with good quality.

"The next release of net.db addresses many of the things currently missing, as there is better customization, business logic, master-detail pages, multi-language support, ISP deployment, and much more. Adding a possibility to integrate business logic developed in CTD is one of the most interesting things and might help to move net.db from a simple Web publishing tool to a powerful Web development tool. However the easy Web publishing features will stay in place, so it will fit both needs in the future. Business logic integration is done via COM, so it will perfectly integrate with CTD 2.0. It will give you access to CGI parameters, page variables, and cookies and will allow you to dynamically create objects on a net.db page out of your business logic implementation."

Martin has quite a bit of additional commentary on topics like SAP R/3, the Ice Tea Group's products, and true n-tier architecture. You can read his full report on the Web at http://www.byte.at/conference_london_en.html.

Continues on page 5

All the Taste Mark Hunter, **Half the Fat** Dian Schaffhauser, **New Minty Flavor** Shelley Doyle, **Delicious Hot or Cold** Paul Gould, **Cleans Your Teeth While You Chew** Mocha

Centura Pro (ISSN: 1093-2100) is published monthly (12 times per year) by Pro Publishing, PO Box 2399, Nevada City, CA 95959.

POSTMASTER: Send address changes to *Centura Pro*, PO Box 2399, Nevada City, CA 95959.

Subscription information: To order, call Pro Publishing at 530-265-4082. Cost of domestic subscriptions: 12 issues, $119; Canada: 12 issues, $129. Other countries: 12 issues, $139. Ask about source code disk pricing. Individual issues cost $15. All funds must be in U.S. currency.

Call Centura Software Corp. at 650-596-3400.

If you have questions, ideas for bribing authors, or would just love to chat about what you're doing with Centura products, contact us via one of the means at right.

Contact Us

Centura Pro on the Web
http://www.ProPublishing.com

Editorial Department
Phone: 818-249-1364
Fax: 603-792-2774
E-mail: huntersoftware@netscape.net

Subscription Services
Phone: 530-265-4082
Fax: 530-265-0368
E-mail: shelley@propublishing.com

Mail
Pro Publishing
PO Box 2399
Nevada City, CA 95959

SAL Gets Smart

Continued from page 1

- Display of function descriptions as multi-line tooltips.
- The ability to automatically scroll the outline to a selected function or member.
- Naming conventions for private functions and members can be configured as appropriate.

As a special bonus, it's possible to move a single selected child item of a window in design mode with the cursor keys (a greatly missed feature).

How is it similar to, and different from, IntelliSense in Visual Studio?

I guess we've imitated almost 100 percent of the functionality IntelliSense offers. Plus we provide autocorrection for some objects and multi-line tooltips that are extremely useful when dealing with big parameter lists or function descriptions.

What tools did you use to write it?

IntelliSal is written with Visual C++ 6.0 and the Centura C++ CDK.

Were you able to include all the features you wanted, or were there some that were too difficult to accomplish?

So far I've implemented all major features I was thinking of. There have been some difficulties in various areas—especially the limits in the CDK notification interface—but all obstacles could be overcome (thanks to the ITG´s assistance).

What's next for IntelliSal?

I plan to work on a deluxe version of IntelliSal that should include, among other features, an easy to use bookmark functionality, because navigation in CTD is quite difficult (especially when mainly working with the keyboard). **CP**

Trying It Out

I liked what I saw when I evaluated IntelliSal. It can run simultaneously with the Coding Assistant, but you may often prefer IntelliSal because of its greater precision. For example, when I wanted to make a call to a function of a UDV from outside the scope of that UDV, Coding Assistant was unable to display those functions from my current outline context. But as soon as I typed the name of the UDV plus a period, IntelliSal popped up all the available functions and variables immediately (see **Figure 1**). Also, the Coding Assistant tends to have very large lists, while IntelliSal pays more attention to the current scope to display smaller, but more relevant, lists.

I usually work with multiple views in the outliner, and IntelliSal has a major advantage in this case. If you're working in a view other than the main view, and the IntelliSal list of functions and variables is visible, you can select a list entry, then press F10. When you do, the main view will scroll to that function, while your position in the other view remains unchanged. This can be much faster than locating the function manually in the Explorer-style pane of the main view—the bigger the app, the bigger the advantage. In fact, Frank enhanced the F10 feature during my testing; now you can highlight any text in the outline (even if the IntelliSal popup isn't present), press F10, and IntelliSal will attempt to scroll to a function matching your selection. Very nice!

And finally, like everyone else who sees it, I love the little "bonus" feature: When you're working in the layout window, you can click on a single control, then use the arrow keys to move it up, down, right, or left. Long overdue and very welcome.

IntelliSal costs $190 US per seat, $170 per seat in quantities over five. It's available from the Ice Tea Group Web site, www.iceteagroup.com. The author, Frank Böttcher, is planning to release a second tool, called IntelliSearch, soon, which adds an advanced search functionality to CTD, including the support of regular expressions.—*M.H.*

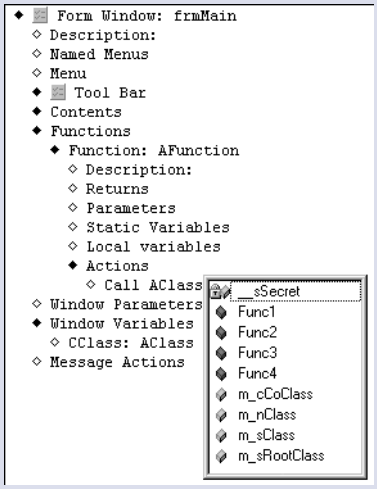

Figure 1. IntelliSal monitors every keystroke and pops up context-sensitive assistance.

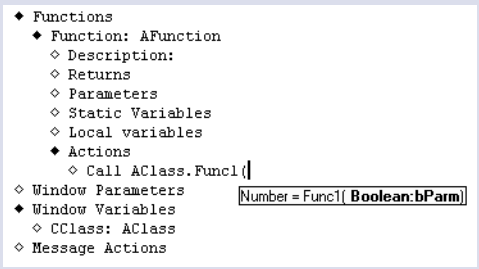

Figure 2. Type a function name and a left parenthesis, and the function prototype appears.

Frank Böttcher has been an independent IT consultant since 1998. Centura came into his life in 1993, and since then he has worked with SQLWindows and SQLBase in various projects including customer information systems, service information systems, and technical research systems. He is particularly fond of the Centura CDK and plans to release more tools soon. He can be reached at frank@iceteagroup.com.

Connecting Legacy Centura Applications to the Web

http://support.sitejet.com
1.888.748.3538 (USA)
1.916.214.3100 (Outside USA)

JetObject technology enables companies to quickly and easily build cross-platform, truly interactive Web sites. SiteJet provides an easy-to-use and flexible server-based solution for creating enterprise portals, e-commerce sites, intranets and extranets that easily integrate with existing IT infrastructure.

JetObject™ Server

- The JetObject Server provides dynamic page generation, user authentication, session management and state management
- The JetObject Server includes an embedded object store used to manage persistent objects
- JetObject messaging provides seamless integration with applications developed using Centura Team Developer and other client/server tools

JetObject™ Pages

- JetObject Pages technology can generate interactive, viewer-specific information
- Content assembled into a Web page can easily be drawn from legacy data sources
- JetObject Pages separate content generation and page layout
- JetObjects facilitate Web site customization
- JetObjects promote the reuse of design elements
- Access lists can be used to limit access to any JetObject page

"This highly synergistic juxtaposition of JetObjects and Centura Team Developer should appeal to any company that is looking to connect their systems to the Web," said Robert Montgomery, President of SiteJet, Inc. "These companies will find that JetObject Pages technology significantly reduces the time and effort needed to build and manage their Web sites."

For additional information on SiteJet's products and services, mail or fax the following form to:
SiteJet, Inc., PO Box 560, Rancho Murieta, CA 95683. Fax: **916-314-3112**

Name: _____ **Position:** _____

Company: _____

Address: _____

_____

Phone: _____ **Fax:** _____

Email: _____

Radio London . . .

Continued from page 2

Idea Engineering also attended the conference, and several reports are available on their Web site at http://www.idea-engineering.co.uk. Some excerpts from their stuff:

> "Charles McLough, a Senior Consulting Engineer at Centura, explained how OLE DB is being treated as the evolution of ODBC: 'OLE DB is a COM-based solution to Universal Data Access.'
>
> "Easier to configure than ODBC and with improved functionality, it's a key element in the Microsoft Universal Data Access strategy. Being a participant in this, Centura is currently developing an OLE DB provider for SQLBase (currently in beta—for more details see the Products section of Centura's Web site).
>
> "In an informative and well-received session, Charles went on to demonstrate the improved performance of OLE DB over ODBC, connecting faster to a database with a simple application.
>
> "Overall, while comfort can be derived from the on-going commitment to development tools, there was a feeling that much of what was being promised had been served up before. This is not meant to undermine the efforts of the development team—it's good to see Centura honoring earlier commitments—but one feels compelled to compare the in-house efforts with those of third-party developers. Centura's 'The Way Forward' presentation doesn't currently leverage such third-party innovations."

The Ice Tea Group got into the act, too. You can read their complete comments at www.iceteagroup.com. Here's an excerpt:

> "We have seen a running sample of Matterhorn where CTD is able to expose itself as a COM Automation Server. The outline has been extended to hold the required new properties, and they've developed a nice wizard for it. Unfortunately, visual COM objects can't be exposed. Though we would like to see many more improvements, we think that in this world of rapid changes Matterhorn will be the first step in the right direction."

In addition to these public comments, others expressed private opinions that were generally positive, approving of Centura's tangible demonstrations of progress with various products.

Learn more about the London conference by viewing slides from the presentations online at www.centurasoft.com/company/events/euro99/index.html.

Forget Y2K; here comes Form 10-K

Form 10-K is a detailed annual report prepared by publicly-traded companies in the U.S. for their shareholders. It's carefully worded to show the potential opportunities for the company while also disclosing the potential risks. It's not a perfect way to understand the company's thinking, but it's better than simply browsing a Web site.

In the most recent 10-K Centura emphasizes that they're looking to the small, mobile, and embedded marketplace for their future growth. Database routers and 4GL tools are mentioned throughout the report, but nearly always with less emphasis than SME products. The report notes that in 1998 product revenues decreased 18 percent from 1997, driven essentially by decreased sales of Centura Team Developer, and indicates that the decreases in tool sales were "due primarily to increased competition." CTD accounted for 21 percent of net product revenues in 1998. SQLBase represented 70 percent of net product revenue in 1998. So one could speculate that Centura's current product priorities are guided by the changing revenue mix.

Centura reports that it has entered into source code escrow agreements with some customers. These agreements grant the customers the right to use Centura's source code under certain conditions, such as Centura ceasing to do business. Such agreements seem to be more widespread throughout the software industry in recent years. In addition, Centura "has, in certain cases, licensed its source code to customers for specific uses." Hmm—take note, you third-party tool builders.

Research and development are the heart of a software company. Centura reports the following:

Year	R&D Expense	% of Net Revenue
1996	$11.0MM	17%
1997	$9.7MM	17%
1998	$7.9MM	15%

Currently there are 43 employees in R&D, 8 in operations and manufacturing, 94 in sales and marketing, and 35 in MIS, finance, and administration. Not part of the 10-K report, but published elsewhere, is the fact that Centura's pay levels are very competitive, even in the relatively expensive Silicon Valley region.

Occasionally, on newsgroups, individuals comment on Centura's stock price, wishing it would go up. The 10-K discusses a few things that would happen if it did go up. As a result of past restructuring and other transactions, there are several sets of warrants and options that can be exercised to purchase newly issued Centura stock at specific prices. This would result in "dilution" of the existing shares on the market. Some of the more significant price points:

Company	No. of shares	Exercise price	Expiration Date
Computer Associates	500,000	$1.906	2/27/2004
"Private Placement"	582,548	$1.25	2/27/2003
Rochon Capital Group	354,717	$2.12	2/27/2003
Newport Acquisition Co.	300,000	$2.09	n/a
Pacific Business Funding	90,000	$2.094	6/30/2002
Sand Hill Capital	10,000	$2.094	6/30/2002

And, of course, Centura's common stock was diluted by 5.8 million shares to complete the recent acquisition of Raima Corp. **CP**

User-Defined Sorts for UDVs

R.J. David Burke

The VisArraySort hack for sorting UDVs has some pretty serious disadvantages. In this article, the author shows a more reliable way to sort them, a way that is also more powerful and polymorphic.

To make sorting simpler, the Visual Toolchest library includes VisArraySort, a handy function for sorting arrays of the SAL primitive data types: Strings, Numbers, and Date/Times. But there's no direct support in SAL or the Visual Toolchest for sorting complex data types built using functional classes. In this article I present a way to sort arrays of User Defined Variables (UDVs), instances of functional classes.

A word of caution

Some SAL developers have reported, in the Centura newsgroups, success in using VisArraySort to sort arrays of UDVs. However, I caution you against using this function to sort UDV arrays. For it to work, the sort key has to be the first instance variable of the Functional Class. You need to provide the data type of the sort key as a parameter to the function call. If the class is derived from another Functional Class, then the first instance variable of the root base class is the sort key. This has various bad implications such as violating encapsulation and needing to know more about the class definition and implementation than should be necessary.

SQLWindows 16/32

I've also found that VisArraySort isn't always reliable with UDV arrays. In some class definitions, VisArraySort will exchange the sort key instance variable, but not the other instance variables of the UDV.

Given these issues, it seems desirable to explore and use other methods for sorting UDV arrays. Another advantage of implementing one's own method of sorting is that you can sort on non-standard sorting criteria, as you'll see in the example to come.

Elements of sorting

There are various sorting algorithms: the insertion sort, the selection sort, the shell sort, the merge sort, and the quick sort are some of the more popular implementations. What they all have in common is that they are essentially a series of comparisons and exchanges. In order to sort UDV arrays, we'll need ways to compare UDVs (analogous to less than/greater than relationships), and we'll need a way to exchange elements of UDV arrays.

Comparing UDVs

For primitive data types, you have a variety of comparison, or relational, operators: less than (<), less than or equal (<=), greater than (>), greater than or equal (>=), not equal (!=), and equal (=). However, when implementing sorting algorithms, you can usually get by with just one or two operators. I've already decided on implementing a quick sort algorithm that needs "less than" and "greater than" operators. On a personal note, I find the terms "less than" and "greater than" too closely tied to the domains of numeric data, so I use the terms "precedes" and "follows" instead.

SAL doesn't support defining new operators, but you can effectively use functions instead. For my purposes, I'll define functions called Precedes and Follows that return a Boolean TRUE or FALSE as appropriate.

One choice would be to define Precedes and Follows as global functions that take two UDVs as parameters and compare the UDVs. A more object-oriented approach is to define Precedes and Follows as functions in the UDV Functional Class. These functions can take just one UDV parameter and compare the parameter relative to itself (the UDV object).

In order to explain and clarify, I'll take this opportunity to introduce an example functional class: Color, as shown in **Listing 1**.

Listing 1. The Color Functional Class.

```
◆ Functional Class: Color
  ◇ Description:
  ◇ Derived From
  ◇ Class Variables
  ◆ Instance Variables
    ◇ String: i_sColor
    ◇ Number: i_nData
  ◆ Functions
    ◆ Function: SetColor
    ◆ Function: GetColor
    ◆ Function: SetData
```

```
◆ Function: GetData
◆ Function: Precedes
  ◇ Description:
  ◆ Returns
    ◇ Boolean:
  ◆ Parameters
    ◇ Color: o
  ◇ Static Variables
  ◆ Local variables
    ◇ Boolean: bRet
    ◇ String: s
  ◆ Actions
    ◇ Set s = o.GetColor(  )
    ◆ If i_sColor = 'Black'
      ◆ If s = 'Black'
        ◇ Set bRet = FALSE
      ◆ Else
        ◇ Set bRet = TRUE
    ◆ Else If i_sColor = 'Burgundy'
      ◆ If s = 'Black' or s = 'Burgundy'
        ◇ Set bRet = FALSE
      ◆ Else
        ◇ Set bRet = TRUE
    .
    .
    .
    ◆ Else If i_sColor = 'White'
      ◇ Set bRet = FALSE
    ◇ Return bRet
◆ Function: Follows
  ◇ Description:
  ◆ Returns
    ◇ Boolean:
  ◆ Parameters
    ◇ Color: o
  ◇ Static Variables
  ◆ Local variables
    ◇ Boolean: bRet
    ◇ String: s
  ◆ Actions
    ◇ Set s = o.GetColor(  )
    ◆ If i_sColor = 'Black'
      ◇ Set bRet = FALSE
    ◆ Else If i_sColor = 'Burgundy'
      ◆ If s = 'Black'
        ◇ Set bRet = TRUE
      ◆ Else
        ◇ Set bRet = FALSE
    .
    .
    .
    ◆ Else If i_sColor = 'White'
      ◆ If s = 'White'
        ◇ Set bRet = FALSE
      ◆ Else
        ◇ Set bRet = TRUE
    ◇ Return bRet
◆ Function: Clone
  ◇ Description:
  ◆ Returns
    ◇ Boolean:
  ◆ Parameters
    ◇ Color: c
  ◇ Static Variables
  ◇ Local variables
  ◆ Actions
    ◇ Call SetColor( c.GetColor(  ) )
    ◇ Call SetData( c.GetData(  ) )
```

The Color Functional Class is used to implement a new data type: the domain of colors. As an implementation detail, we store the color value as an instance variable string that contains the plain text color word such as "Red" or "DarkGreen". There's a fixed set of permissible values for the Color class, which is evident from examining the full source code. The i_nData instance variable is simply provided to augment the example code, fleshing out the functional class. The Color class includes basic accessor functions to get and set the instance variables.

For this example, sorting instances of the Color class means arranging the instances in RGB order (as opposed to sorting the instances alphabetically on i_sColor. This means that the Precedes and Follows functions must "understand" the relative RGB order relationships between the possible domain values for the Color class.

Let's look more closely at the Precedes function. It takes another Color object as a parameter. Remember that when calling Precedes it will be qualified by an instance of the Color class. This means that the current instance is compared to the parameter. TRUE is returned if the current instance precedes the parameter; otherwise, FALSE is returned. Follows behaves similarly. The bulk of code for Precedes and Follows is tedious and is omitted from Listing 1, but the general idea is conveyed.

Exchanging UDVs

The Clone function in Listing 1 is used to support exchanging UDV instances. The typical approach in SAL is to replicate each instance variable from the source to the target (the current instance). As I'll show you later in this article, the logical implementation of an exchange is a classic three-step operation that uses temporary storage to save one of the instances as they're swapped.

The amazing quick sort

The quick sort, first introduced by C.A.R. Hoare, is the classic, high-performance algorithm for sorting data. In this particular implementation, shown in **Listing 2**, there are two notable features: 1) the Precedes and Follows functions are called, respectively, instead of using "<" and ">" operators; and 2) the quick sort algorithm is split between the QuickSort function, which is the entry point for sorting, and the Split function, which is recursively called as needed.

Listing 2. A quick sort for UDV arrays.

```
◆ Function: QuickSort
  ◇ Description:
  ◆ Returns
    ◇ Boolean:
  ◆ Parameters
    ◇ Color: p_colors[*]
    ◇ Number: pnFirstItem
    ◇ Number: pnLastItem
  ◇ Static Variables
  ◆ Local variables
    ◇ Boolean: bRetVal
    ◇ Number: nSplitPoint1
    ◇ Number: nSplitPoint2
  ◆ Actions
    ◇ Set bRetVal = TRUE
    ◆ If pnFirstItem < pnLastItem
      ◇ Call Split( p_colors, pnFirstItem,
        pnLastItem, nSplitPoint1, nSplitPoint2 )
      ◆ If nSplitPoint1 < pnLastItem
        ◇ Call QuickSort( p_colors,
          nSplitPoint1, pnLastItem )
      ◆ If pnFirstItem < nSplitPoint2
        ◇ Call QuickSort( p_colors,
          pnFirstItem, nSplitPoint2 )
    ◇ Return bRetVal
◆ Function: Split
```

```
◇ Description:
◆ Returns
  ◇ Boolean:
◆ Parameters
  ◇ Color: p_colors[*]
  ◇ Number: pnFirstItem
  ◇ Number: pnLastItem
  ◇ Receive Number: pnSplitPoint1
  ◇ Receive Number: pnSplitPoint2
◇ Static Variables
◆ Local variables
  ◇ Number: nRight
  ◇ Number: nLeft
  ◇ Color: pivot
  ◇ Color: temp
◆ Actions
  ◇ Call pivot.Clone( p_colors[
    SalNumberRound( ( pnFirstItem +
    pnLastItem ) / 2 - 0.5 ) ] )
  ◇ Set nRight = pnFirstItem
  ◇ Set nLeft = pnLastItem
  ◆ Loop
    ◆ While p_colors[ nRight ].Precedes( pivot )
      ◇ Set nRight = nRight + 1
    ◆ While p_colors[ nLeft ].Follows( pivot )
      ◇ Set nLeft = nLeft - 1
    ◆ If nRight <= nLeft
      ◇ Call temp.Clone( p_colors[ nRight ] )
      ◇ Call p_colors[ nRight ].Clone(
        p_colors[ nLeft ] )
      ◇Call p_colors[ nLeft ].Clone( temp )
      ◇ Set nRight = nRight + 1
      ◇ Set nLeft = nLeft - 1
    ◆ If nRight > nLeft
      ◇ Break
  ◇ Set pnSplitPoint1 = nRight
  ◇ Set pnSplitPoint2 = nLeft
```

So there you have it. In my environment, the sample program (from file UDVSORT.ZIP) takes about 45 seconds, on the average, to sort an array of 2,000 Color objects. And most of that time is in the Precedes or Follows functions. The sorting code itself is reasonably efficient.

Making it work for you

With a little adaptation, the above code should fit nicely into your environment. To start, you need to add Precedes, Follows, and Clone methods to your Functional Class. Next, you'll add the above functions for sorting, modifying them as required to handle your particular functional class. Then you're off and running! **CP**

Download file UDVSORT.ZIP from this issue's Table of Contents at www.ProPublishing.com or find it on this month's Companion Disk.

R.J. David Burke is a Senior Consulting Engineer with Centura Software Corp. He can be contacted through the editorial staff of *Centura Pro*.

Keep Track with SQLRouter

Centura Tip!

Centura Software Technical Support—An activity log can help you troubleshoot problems and provide important information to Centura Software Technical Support if you need engineering help to resolve your problem. The log activity generates a log that captures most of the important elements of SQLRouter processing and can help Technical Support engineers resolve issues that involve Centura connectivity.

To invoke the SQLRouter activity log, you must have the following section and keyword specification in your SQL.INI file:

```
[win32client.<dll_name>]
log=<fully_qualified_path_name>
```

. . . where the dll_name is the first three letters of the comdll entry plus "32." For example, Oracle's section name is win32client.ora32.

You can specify any path or filename for the keyword. If a file already exists, it's overwritten. Once the keyword is specified, logging is performed automatically with each SQLRouter initialization. You can apply one or more of the following (case-sensitive) options to customize the output:

/Bx	exclude display of bind variable values
/Fa	append to existing file
/FD	display fetch data
/Ld	display LONG data read/written
/Tx	exclude display of timestamps

SQLRouter

The following example results in a SQLRouter/ODBC log file that doesn't include timestamps on entries (conserving space) and displays the input and output of all LONG (in other words, TEXT or IMAGE) data items.

```
[win32client.odb32]
log=c:\centura\odbrtr.log /Tx /Ld
```

In most cases, the default (no options) is appropriate and sufficient.

The Type-Ahead Trick

Rajeev Mitra

The first issue of *Gupta Developer*, predecessor of *Centura Pro*, contained a hint about Quicken-style combo boxes. In this article, Mitra updates and extends the technique. Use type-ahead combos for a crowd-pleasing addition to your user interface.

If you've used Quicken, you've already seen the type-ahead combo box. As you enter keystrokes in a type-ahead combo box, the first match from the list is selected. The text you have typed so far is unselected, and the rest of the text appears as selected. I will show you how easy it is to implement this functionality in your class library.

The functionality is implemented in the SAM_AnyEdit message. I find the current position of the cursor using the CB_GETEDITSEL message (0x0140). This message returns the starting and ending positions as a 32-bit return of SalSendMsg. The low word is the starting position and the high word is the ending position. You can obtain the low and high word using functions SalNumberLow and SalNumberHigh. I store the return value in a variable (nSelection) for future use. I also store the current combo box text (MyValue) in a variable, sCurValue. Then, I look through the list using SalListSelectString for the combo box text. If the value isn't found (the return is LB_ERR), I restore the combo box text to its original value (sCurValue) and re-position the cursor using message CB_SETEDITSEL (0x0142) and nSelection. If the value is found, I select the matched list-box entry using SalListSetSelect. I recalculate the new selection by combining low word of nSelection and -1 into a long and using this value in message CB_SETEDITSEL. That was pretty easy, wasn't it? The code is shown in **Listing 1**.

SQLWindows 16/32

Listing 1. The core logic of a typeahead combo box.

```
◇ Set nSelection = SalSendMsg(hWndItem,
  CB_GETEDITSEL, 0, 0)
◇ Set sCurValue = MyValue
◇ Set nIndex =
  SalListSelectString(hWndItem, -1, sCurValue)
◆ If nIndex != LB_Err
  ◇ Call SalListSetSelect(hWndItem, nIndex)
  ◇ Call SalSendMsg(hWndItem, CB_SETEDITSEL,
    0, SalNumberLow(nSelection) + (0x10000 * -1))
◆ Else
  ◇ Set MyValue = sCurValue
  ◇ Call SalSendMsg(hWndItem, CB_SETEDITSEL,
    0, nSelection)
◇ Call SalSetFieldEdit(hWndItem, TRUE)
```

Are we done yet?

Not quite! There are a couple of things that we need to take care of. The first thing is the Backspace key. Backspace should work like this: If text from position n to the end of the text is selected, Backspace should select text from position n-1 to the end of the text. If n happens to be 0 or 1, the combo box text should be cleared. If the selection is somewhere in the middle of the string, just remove the selected part.

I trap the Backspace key in the WM_KEYDOWN message. After processing the message, I return zero to ensure that CTD doesn't subsequently process the message. I thought it would be very easy to implement. It turned out to be slightly difficult. It seems that even on returning zero, CTD still sends the SAM_AnyEdit message anyway. This required a little different programming for the case when the selection is somewhere in the middle of the string. In this case, I first see if anything is selected. If it is, I remove all but one character of the selected string. Also, I set a Boolean to TRUE. If this Boolean is TRUE, I just return from the original code shown in Listing 1, without executing it. The code for Backspace looks like this:

```
◆ If SalStrLength(MyValue) > 0
  ◇ Set nSelection = SalSendMsg(hWndItem,
    CB_GETEDITSEL, 0, 0)
  ◆ If SalNumberHigh(nSelection) =
    SalStrLength(MyValue)
    ◆ If SalNumberLow(nSelection) =
      0 OR SalNumberLow(nSelection) = 1
      ◇ Set MyValue = STRING_Null
      ◇ Call SalSetFieldEdit(hWndItem, TRUE)
    ◆ Else If SalNumberLow(nSelection) !=
      SalNumberHigh(nSelection)
      ◇ Call SalSendMsg(hWndItem, CB_SETEDITSEL,
        0, SalNumberLow(nSelection)-1+(0x10000*-1))
  ◆ Else
    ◆ If SalNumberLow(nSelection) !=
      SalNumberHigh(nSelection)
      ◇ Set MyValue = SalStrLeftX(MyValue,
        SalNumberLow(nSelection) + 1) ||
          SalStrRightX(MyValue, SalStrLength(MyValue) -
                SalNumberHigh(nSelection))
      ◇ Call SalSendMsg(hWndItem, CB_SETEDITSEL,
        0, SalNumberLow(nSelection)+1 +
        (SalNumberLow(nSelection)+1)*0x10000))
    ◇ Set iv_bInBackspace = TRUE
  ◇ Call SalSetFieldEdit(hWndItem, TRUE)
◇ Return FALSE
```

The second thing to take care of is the Delete key. In case of Delete, I just delete the selected string. As in Backspace, Delete is implemented by trapping the WM_KEYDOWN message and returning zero when the processing is over. Luckily, returning zero in Delete doesn't lead to SAM_AnyEdit being sent. Hence, Delete was relatively easy to implement. The code looks like this:

```
◇ Set nSelection =
  SalSendMsg(hWndItem, CB_GETEDITSEL, 0, 0)
◇ Set MyValue = SalStrLeftX(MyValue,
  SalNumberLow(nSelection)) ||
    SalStrRightX(MyValue, SalStrLength(MyValue) -
      SalNumberHigh(nSelection))
◇ Call SalSendMsg(hWndItem, CB_SETEDITSEL, 0,
◇ SalNumberLow(nSelection)+
  (SalNumberLow(nSelection)*0x10000))
◇ Call SalSetFieldEdit(hWndItem, TRUE)
```

One last thing

Another thing to consider is what to do if the string typed in doesn't exist in the list. I decided to provide a function for the user of the class, defining whether new values are allowed or not. If new values aren't allowed, then on SAM_KillFocus, I check the string typed in against the combo box entries; if it isn't found, I simply set the combo box text to STRING_Null. The code looks like this:

```
◆ If NOT iv_bAllowNewValues  ! only allowed
                ! to select from list
  ◆ If SalListSelectString (hWndItem, -1, MyValue) =
    LB_Err
    ◇ Set MyValue = STRING_Null
```

The bug

Yes, that's right; let's talk about the "bug." During my testing, I found that if I drop down the list box, type to select an entry, and then tab out, the combo box re-selects the entry that was selected before setting focus in the combo box. I tried various things, but nothing would fix the problem. Then I called Centura Tech Support. They researched the problem for me and told me that it's a Windows bug. One solution they suggested was to somehow send a down-arrow key on SAM_KillFocus. That also didn't work to my satisfaction. If you can come up with a better solution, please share it with other *Centura Pro* readers and me. **CP**

All the logic shown above is available in the sample source code file TypeAhead.APT, available on this month's Companion Disk, or from this issue's Table of Contents at www.ProPublishing.com.

Rajeev Mitra works as a developer in New York's financial world. Contact him at r.mitra@worldnet.att.net or rajeev.mitra@mbia.com.

Let the Dragging Begin

Centura Tip!

R.J. David Burke—To initiate drag mode in SAL, the typical approach is to use code similar to the following:

```
◆ On SAM_DragCanAutoStart
  ◇ Return TRUE
```

When the mouse is positioned over a control (like a list box) and the left mouse button is held down for a short time, the runtime system automatically sends a SAM_DragCanAutoStart message to the control. Responding TRUE to this message automatically initiates drag mode.

SQLWindows uses this approach to differentiate between selecting an object or data and dragging that object or data. However, some developers prefer a drag and drop model similar to that in Windows Explorer. In the Explorer drag and drop model, drag mode is initiated when the left mouse button is pressed down and the mouse has been moved at least two pixels from the left button down point. To accomplish this behavior in SQLWindows, use the following code:

```
◆ On WM_LBUTTONDOWN
  ◇ Set nDownXPos = SalNumberLow( lParam )
  ◇ Set nDownYPos = SalNumberHigh( lParam )
  ◇ Set bLButtonDown = TRUE
◆ On WM_LBUTTONUP
  ◇ Set bLButtonDown = FALSE
◆ On WM_MOUSEMOVE
  ◆ If bLButtonDown
    ◇ Set nCurrentXPos = SalNumberLow( lParam )
    ◇ Set nCurrentYPos = SalNumberHigh( lParam )
    ◆ If SalNumberAbs( nCurrentXPos - nDownXPos ) > 2
      OR SalNumberAbs( nCurrentYPos - nDownYPos ) > 2
      ◇ Call SalSendMsg( hWndItem,
        WM_LBUTTONUP, wParam, lParam )
      ◇ Call SalDragDropStart( hWndItem )
```

You'll also need to define the following constants:

```
◇ Number: WM_MOUSEMOVE = 0x0200
◇ Number: WM_LBUTTONDOWN = 0x0201
◇ Number: WM_LBUTTONUP = 0x0202
```

For an OOP approach, place the above code in a general window class and derive appropriate classes as needed in your application.

The downloadable file listed below demonstrates the technique presented in this tip.

Download file QuickDragDrop.app from this issue's Table of Contents at www.ProPublishing.com or find it on this month's Companion Disk.

R. J. David Burke is a Senior Consulting Engineer with Centura Software Corp.

10 Steps to "Page X of Y" Reports

R.J. David Burke

A long outstanding feature request of Report Builder (and ReportWindows) is support for "Page X of Y" reports. While this feature request is still "in the queue," here's a workaround to accomplish the same from within SAL.

The ability to print reports with pages numbered like "Page 1 of 3," "Page 2 of 3," "Page 3 of 3" can be desirable in many scenarios; it adds to the "user-friendliness" of the reports and lets readers of the report feel confident that they have all the pages.

Back in the heyday of dot-matrix printers, this kind of page numbering was easy, because line counts were predictable (66 lines per page on 8.5- x 11-inch paper). But the features of modern report writers, including fonts, graphics, conditional printing, word wrap with multiple lines, etc. complicates this kind of page numbering.

The most expedient way to determine the page count of a report is to generate the report twice (two-pass reporting). Use the first pass to determine the page count and then discard it. Pass the page count on to the second pass. An important issue in this approach is ensuring that the same data is used in each pass to generate the report. Various strategies can be used, such as caching the data on the client (with a table window, for example) or using a suitable isolation level on the server (RR or RO with SQLBase) if the report is driven from data in a database. Use whatever is appropriate for your application, but make sure that the same data is used for both passes.

Report Builder

You might think that you could use Report Builder's Report Totals features with two passes, but you'd soon find out that they're not suitable, because they only fetch the data once. All the report data is buffered by the report engine and run through twice, but not fetched twice.

So you'll need to generate the report twice, under the direction of the application. A few years ago, Charity Silkebakken published a technique for doing this. She would call SalReportPrint twice, the first time specifying a printer defined as a NUL DOS device so it would be discarded. Then the second pass would print to a normal printer. The problem with this approach is that it required special configuration of the end-user's machine that materialized as a significant deployment issue.

The approach detailed here is simpler in that the first pass is done with SalReportView so there's no need to configure a NUL printer. So here are the details:

1. First you'll want to open your report template (.qrp file), and add a report variable that will contain the page count. For example, I've created a report variable call pagecount. Assign this report variable to an appropriate field in the report. For example, on the page footer, you could have:

   "Page " PageNumber( ) " of " pagecount

2. Then you'll want to initialize a couple of useful variables:

```
◇ Set bFirstPass = TRUE
◇ Set bFirstPage = TRUE
```

   bFirstPass remains TRUE until the first pass of generating the report is complete. bFirstPage remains TRUE while the first page is generated in Report Builder.

3. You'll need to create an empty form window to contain the report. Letting SalReportView create its own window for displaying the report isn't desirable because you'll want to have more control over the container window.

```
◇ Set hWndContainer = SalCreateWindow( __frmReport,
hWndNULL )
```

4. You don't want the user to see the container window used for the first pass of the report, so the following message handler can be used with __frmReport:

```
◆ On SAM_Create
  ◇ Call SalHideWindow( hWndForm )
```

Next, you generate the first pass of the report in the hidden container window, using SalReportView:

```
◇ Set hWndReport = SalReportView( hWndForm,
        hWndContainer, sReportTemplate,
        sBindVariables, sInputItems, nErr )
```

5. Now you can just wait until the first page is rendered in the hidden container window. The key thing here is to properly handle the incoming message—specifically, receiving SAM_ReportNotify with an lParam value of RPT_BeforePageFooter on the first page.

```
◆ On SAM_ReportNotify
  ◆ If bFirstPass AND bFirstPage AND lParam =
    RPT_BeforePageFooter
    ◇ Set bFirstPage = FALSE
    ◇ Call SalPostMsg( hWndForm, PM_LastPage, 0, 0 )
```

The PM_LastPage message is posted instead of sent so that the report engine has some "breathing room" to handle the next few messages before going to the last page.

6. The next step is to "drive" the report engine to go to the last page of the report. This can be done programmatically using SalReportCmd with RPT_CmdLastPage.

```
◆ On PM_LastPage
  ◇ Call SalReportCmd( hWndReport, RPT_CmdLastPage )
  ◇ Call SalPostMsg( hWndForm, PM_CloseReport, 0, 0 )
```

The PM_CloseReport message is also posted. This lets the report engine do its work to get to the last page.

7. Back to the SAM_ReportNotify message handler introduced in step 5. You'll need to add another SAM_ReportNotify test to count pages:

```
◆ On SAM_ReportNotify
    .
    .
    .
  ◆ If bFirstPass AND lParam = RPT_BeforePageHeader
    ◇ Set nPageCount = nPageCount + 1
```

8. Once the last page is produced, and all the generated pages have been counted, the first pass report can be closed:

```
◆ On PM_CloseReport
  ◇ Call SalReportClose( hWndReport )
  ◇ Call SalDestroyWindow( hWndContainer )
  ◇ Set bFirstPass = FALSE
```

9. Now you can do the second pass report generation using SalReportView, SalReportPrint, or SalReportPrintToFile:

```
◇ Set hWndReport = SalReportView( hWndForm,
    hWndNULL, sReportTemplate,
    sBindVariables, sInputItems, nErr )
```

10. When the second pass report starts, you can use SalReportSetNumberVar to set the pagecount report variable:

```
◆ On SAM_ReportStart
  ◆ If NOT bFirstPass
    ◇ Call SalReportSetNumberVar( hWndReport,
      'pagecount', nPageCount )
```

And there you have it. Yes, it is a little bit of work, and it's not that elegant, but if you don't have the budget for Crystal Reports or Sqribe, this may do the job. **CP**

Download the code sample, XofY.zip from this issue's Table of Contents at www.ProPublishing.com, or find it on this month's Companion Disk.

R.J. David Burke is a Senior Consulting Engineer with Centura Software Professional Services. You can reach him through the editorial staff of *Centura Pro*.

Better Status for Your Functions *Centura* Tip!

Rajeev Mitra—Here's a way to make your class functions easier to use. When you're in an action block section of an outline, SQLWindows/32 shows a list of available class functions in the Coding Assistant. (You can make coding assistant visible by pressing Alt-2 or by choosing menu Tools | Coding Assistant.) If you select a class function in the coding assistant window, you can see the function description in the status bar of SQLWindows/32 environment. You can take advantage of this feature to make your class functions easier to use. Just specify a meaningful description that starts with the function usage. This will help users of your class who can now just select the function to understand its usage. You can even extend this feature to External Functions defined in your app. As a matter of fact, Visual Toolchest function definitions already do this. Now, the next cool thing Centura can do to help us all is to extend this feature to all SAL functions.

SQLWindows 32

[*Rajeev is right. Function descriptions are being increasingly used in a number of utilities, including DocSal and IntelliSal. There's more reason than ever to write a good function description, instead of leaving it blank—Ed.*]