CenturaPro™

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

Windows API Overdose

Rajeev Mitra with Mark Hunter

You've seen many of the tricks in this article in other tips, or on CompuServe, or elsewhere. But never have so many cool API functions and messages been demonstrated in such a concentrated form. This article is fairly short, but when you download and run the related application (WIN_API.APT), you'll be glued to the screen, fascinated by the interesting, useful, and obscure tricks in this demonstration. You can put them to use in your own apps immediately. Have fun!

Most large SQLWindows applications already make use of Windows API function calls and messages. Messages like EM_SETSEL or WM_CHAR are often used; functions like TrackPopupMenu and ShowWindow are also sometimes found. There are existing APLs that contain definitions of all Windows API constants, including messages. But my app, WIN_API.APT, provides both definitions and usage examples for a very large number of messages and functions. You're sure to find some that you haven't used before.

WIN_API.APT is split into several major functional groups, represented by the buttons on the main form. (See **Figure 1**.) Each button brings up another form, with many related examples.

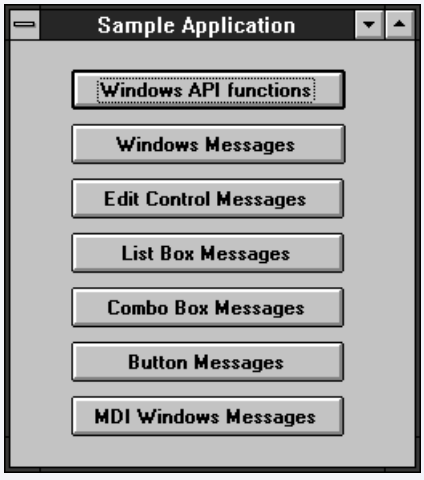

Figure 1. WIN_API divides its API tricks into several function groups.

Some of these secondary forms are almost mind-boggling, but they're cool. Check out the function call examples in **Figure 2**.

Exploring these buttons yields some simple but useful tricks. For example, it would be nice to warn users when system resources are too low to launch further tasks. You can borrow the code that WIN_API uses to display these resources (**Figure 3**).

Often a single function can invoke several different behaviors. For example, you can throw the user out of Windows (one of my personal favorites) in a variety of ways:

Continues on page 4

March 1997

Volume 2, Number 3

PRO Publishing

You've Got Work to Do

Mark Hunter

Each year, *Centura Pro* staffers visit Centura Software headquarters. This year we found a very busy, but more positive, company. Many employees were out of town, assisting with the ForeSite "roadshow." Many others were somewhere in the spectrum between working long hours because they felt they ought to , and working extra because they were excited. This is good news, too; there was a time when the energy level was much lower at the company.

In spite of all this, we found plenty of people to talk to, with plenty to say. Everyone was optimistic about the ability of the company to increase their mindshare and their profits. The hot topic is still ForeSite, and we cover some of the technical features of the product in the short article, "Leap Onto the Web" in this issue. But in our discussions, we kept looking for the special synergy that would make ForeSite and Centura Team Developer into an unbeatable combination. Although ForeSite will do a great job of making CTD applications very scaleable and reliable on the Web, it will also do an equally great job of making Visual Basic, PowerBuilder, or C++ apps scaleable and reliable. In other words, ForeSite is a great product, almost indispensable for large, sophisticated Web sites and intranets.

But part of its greatness is its openness. If Centura tried to take advantage of ForeSite features that weren't available to other vendors, some of that openness would go away, and ForeSite would be less attractive to customers.

One of the challenges facing Centura is how to define ForeSite. It's not a development tool. It's not an HTML editor. When it's working properly, you can't even tell that it's running. It offers two major advantages: the ability to implement massive web sites with huge numbers of visitors, and the ability to provide dynamic content through links to databases, OLE servers, and other applications. Many companies aren't even aware that it exists, but the ones that are seem to love it. See the news item, "On the Web Side," on our Web site for more information on how Centura will position and promote the product (Love those prizes!)

Editor Mark Hunter, **Publisher** Dian Schaffhauser, **Business Manager** Shelley Doyle, **Production Editor** Paul Gould, **Loyal Companions** Ruthie and Mocha

Centura Pro is published monthly (12 times per year) by Pro Publishing, PO Box 18288, Seattle, WA 98118–0288.

POSTMASTER: Send address changes to *Centura Pro*, PO Box 18288, Seattle, WA 98118–0288.

Subscription information: To order, call Pro Publishing at 206-722-0406. 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.

Centura technical support: Call Centura Software Corp. at 415-321-4484.

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.

Send a Love Note

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

Editorial Command Post
Phone: 818-249-1364
Fax: 818-246-0487
E-mail: 71460.3142@compuserve.com

Subscription Headquarters
Phone: 206-722-0406
Fax: 206-760-9026
E-mail: 71333.2142@compuserve.com

Mail
Pro Publishing
PO Box 18288
Seattle, WA 98118-0288

Source Code on CompuServe
GO CENTURA, Library 10

We're a Web company—are you a Web company?

All of this fits nicely into Centura's new focus as a Web company. What does that mean, anyway? They aren't a tools company—or database company—anymore? To understand this focus, remember that Centura management is always asking, "Where are the *new* customers coming from?" Looking at it this way, Centura Team Developer won't create the big growth the company is looking for. No matter how good a tool it is, sales will always be lost to customers who think they can do everything with VB or some other low-cost tool. SQLBase is a good source of income and popular with OEMs, but it's also unlikely to provide huge growth. These products will be improved and will continue to provide modest growth; but Centura Software needed to find a market where they could assume a leadership position quickly. Now they're in the high-end Web deployment business, which complements the corporate message that Centura products can handle anything the enterprise requires.

Centura Team Developer won't create the big growth the company is looking for.

Now the question you must answer: Are your applications going to move to the Web or to an intranet? Centura is quickly perfecting products to make this easy, and if you want to be ready, you must start preparing now. During our visit, we looked at demos of some actual apps running on Web sites. (You can see them too. Just point your browser to www.inside.centurasoft.com, register on that site, and start running demos and downloading betas.)

It became crystal clear that, although Centura is technically capable of putting your *existing* app on the Web through Web Developer, you'll be sorry you did it. Why? Stop and think about the last complex Web site you browsed. It was modal, modal, modal. You moved in a very linear fashion through the tree structure of the site. Each page had a small number of simple choices. If you filled out a form, you didn't find out about any input errors until you pressed the "Submit" button.

This sounds like a typical Web site. But it doesn't sound anything like a typical client/server application. Those applications have rich, complex screens, modeless navigation, instantaneous validity checking, and a large number of specialized child windows. When you put such an app on the Web, much of its richness gets lost, and the remaining objects just look strange and hard to use.

Virtually every direct port of a client/server app has been a failure to those who browsed it, because it didn't act like a Web page. And just imagine what happens when the user clicks the Back or Forward button on their browser! Even Centura's Beerfinder site posts warnings on its pages telling users to click the page's Back button, not the browser's Back button.

So what's the answer? Do you start designing your client/server apps to mimic Web navigation? Do you stay away from editable table window columns because they can't be reproduced in HTML? If you do things like this, you're going to have an unpopular client/server app.

Bite the bullet

To move to the Web, you must face facts: You're going to need two versions of your app: a client/server version and a Web version. The differences will be primarily in the user interface, not the business rules. So, at last, I get to my point: You have work to do. Most of us haven't made a clear separation in our apps between the user interface and the business rules. We must do that now, to make it easy to write Web-style user interfaces for these apps.

Actually, separating the user interface from business rules is an excellent idea anyway, as class library vendors are happy to tell you. Even if your app won't be Web-enabled, this separation makes it easier to move to three-tier architectures. Slowly, slowly, the design principles of client/server and OOP go from being just good ideas to being absolute requirements.

You'll need those good design principles again when you start preparing for Tomahawk. Although it has many new capabilities, including Java byte code generation, Tomahawk is more restrictive about what kinds of external references are allowed in applications. Scoping rules from SQLWindows version 4 are completely disallowed, as are some other forms of reference. Preliminary documentation for Sal97, the language of Tomahawk, has been published on Centura's Web site. You should download it and read it immediately. In addition to learning about many exciting new features, you can read the section on migration to learn what changes to make in your applications.

As we said before, you've got work to do, and it's not too early to get started.

On another note

OK, you long-suffering ReportWindows developers, we heard that a new effort is underway to bring the ReportWindows feature list closer to the popular third-party reporting tools, while keeping the tight integration that everyone likes. We'll follow up on this in future months. There's lots more to talk about, like the new Mambo beta, the Babel beta, and advances in Web Developer . . . **CP**

Windows API Overdose . . .

Continued from page 1

```
Pushbutton: pbRestartWindows
    Message Actions
        On SAM_Click
            Call ExitWindows( EW_RESTARTWINDOWS, 0 )
Pushbutton: pbRestartPC
    Message Actions
        On SAM_Click
            Call ExitWindows( EW_REBOOTSYSTEM, 0 )
Pushbutton: pbExitWindows
    Message Actions
        On SAM_Click
            Call ExitWindows( 0, 0 )
Pushbutton: pbExitWindowsExec
        On SAM_Click
            If SalStrUpper( dfPath, dfPath ) >= 128
                Call SalMessageBox( 'Path length must
                    be less than 128', '', 0 )
                Return FALSE
            If SalStrRightX( dfPath, 4 ) != '.EXE'
                Call SalMessageBox( 'Program to run must
                    be an executable (.exe)', '', 0 )
                Return FALSE
            Call ExitWindowsExec( dfPath, '' )
```

Or see how features of the API can be combined with standard SAL code, in this example from frmWindowLong:

```
Group Box: Create a window that has:
Check Box: cbTitleBar
    Message Actions
        On SAM_User
            If MyValue
                Set nStyle =
                nStyle & ( 0xffffffff - WS_CAPTION )
Check Box: cbDlgFrame
    Message Actions
        On SAM_User
            If MyValue
                Set nStyle =
                nStyle & ( 0xffffffff - WS_DLGFRAME )
Check Box: cbMaximizeBox
    Message Actions
        On SAM_User
            If MyValue
                Set nStyle =
                nStyle & ( 0xffffffff - WS_MAXIMIZEBOX )
Check Box: cbMinimizeBox
    Message Actions
        On SAM_User
            If MyValue
                Set nStyle =
                nStyle & ( 0xffffffff - WS_MINIMIZEBOX )
Check Box: cbSystemMenu
    Message Actions
        On SAM_User
            If MyValue
                Set nStyle =
                nStyle & ( 0xffffffff - WS_SYSMENU )
Check Box: cbSizeable
    Message Actions
        On SAM_User
            If MyValue
                Set nStyle =
                nStyle & ( 0xffffffff - WS_THICKFRAME )
Pushbutton: pbCreateForm
    Message Actions
        On SAM_Click
            Set nStyle = 0xffffffff
            Call SalSendMsgToChildren( hWndForm,
                SAM_User, 0, 0 )
            Set nStyle = nStyle &
                ( 0xffffffff - WS_HSCROLL )
            Set nStyle = nStyle &
                ( 0xffffffff - WS_VSCROLL )
            Set gnStyle = nStyle
            Call SalCreateWindow( frmTest,
                hWndForm, hWndItem, nStyle )
```

Windows API messages are often simpler to use than function calls, and they can be quite powerful too. WIN_API has examples of several groups of messages. You can micro-manage the user interface with edit control messages, as shown in **Figure 4**.

For an example of simplicity and power, try the code to switch a data field between read-only and read-write:

```
Pushbutton: pbReadOnly
    Message Actions
        On SAM_Click
            Call SalSendMsg( dfReadOnly,
                EM_SETREADONLY, TRUE, 0 )
Pushbutton: pbReadWrite
    Message Actions
        On SAM_Click
            Call SalSendMsg( dfReadOnly,
                EM_SETREADONLY, FALSE, 0 )
```

List boxes can be a little difficult to manipulate in SQLWindows. Even Visual Toolchest missed a few features, like searches on partial strings (see **Figure 5**):

```
Pushbutton: pbFindString
    Title: Find string starting with
    Message Actions
        On SAM_Click
            Call SalMessageBox( 'First entry starting
            with prefix ' || dfFindString ||
            ' is ( -1 on error ):
            ' || SalNumberToStrX( SendMsgByString
```

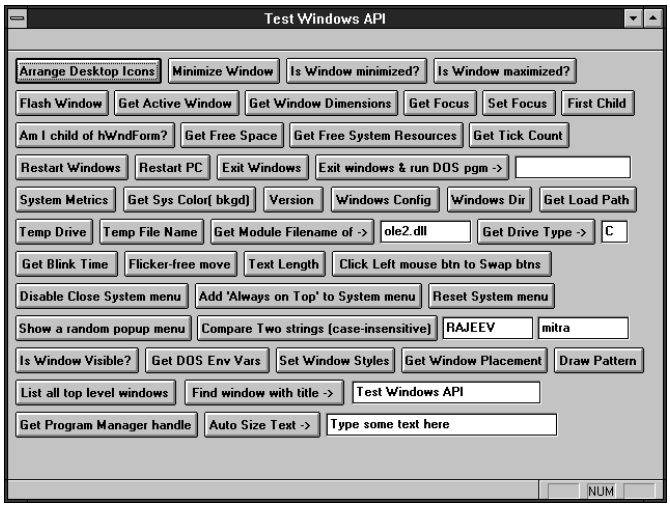

Figure 2. More than you ever wanted to know about API function calls.

Figure 3. You can display system resources or use them to control warning messages in your application.

```
            ( lbSingleSelect, LB_FINDSTRING, -1,
            dfFindString ), 0 ), '', 0 )
```

One of the frustrations of SQLWindows is that you don't always get a horizontal scroll bar on list boxes when it's needed. You can fix that easily:

```
Pushbutton: pbSetHorizBar
    Message Actions
        On SAM_Click
            Call SalGetWindowSize( lbSingleSelect,
                nWidth, nHeight )
            Set nWidth = SalFormUnitsToPixels
                ( lbSingleSelect, nWidth, FALSE )
            Call SalSendMsg( lbSingleSelect,
                LB_SETHORIZONTALEXTENT, nWidth + 25, 0 )
            Call SalInvalidateWindow( lbSingleSelect )
```

. . . and remove the bar just as easily:

```
Pushbutton: pbRemoveHorizBar
    Message Actions
        On SAM_Click
            Call SalGetWindowSize( lbSingleSelect,
                nWidth, nHeight )
            Set nWidth = SalFormUnitsToPixels
                ( lbSingleSelect, nWidth, FALSE )
            Call SalSendMsg( lbSingleSelect,
                LB_SETHORIZONTALEXTENT, nWidth - 25, 0 )
```

What about the Visual Toolchest?

The Visual Toolchest was originally written as a third-party product to provide functionality that wasn't part of SQLWindows. One of the main ways VT delivered this functionality was to serve as a "wrapper" for various Windows API functions. For example, function VisListFindString behaves in the same way as the API message LB_FINDSTRING. However, this message allows both partial and exact matches; VisListFindString allows only exact matches. So the API is still more flexible than Visual Toolchest in some areas.

There are advantages to using Visual Toolchest, however. Simply including VT.APL, or some other VT-related APL, gives you all necessary definitions. If you use the Windows API, you must ensure that the function and message definitions you need are included in your app, and that they don't interfere with, or double-define, definitions that may have been in the app already.

What about the 32-bit API?

The wonderful tricks in this article won't run under Centura Team Developer. Fortunately, a recent tool posted to Centura's CompuServe forum library should help. Look for the file WIZARD or the keyword SDK. It uncompresses to CVT32_WIZ.EXE and several other files that can be used, under CTD 1.0, to automatically convert 16-bit Windows API function calls to 32-bit calls.

Does it work? You'll have to wait until next month for a report on exactly how many of Mitra's tricks converted successfully. Meanwhile, you can download the file and fool around for yourself.

By the way, the mechanism used by CVT32_WIZ.EXE is not limited to converting API calls. Its search-and-replace information is stored in database tables. By adding additional rows to those tables, you can make CVT32_WIZ.EXE do custom edits to your apps for your own functions. Well worth a look!

It never ends

WIN_API.APT is the most comprehensive single example of what can be done by using the Windows API. But there are many other functions and messages not explored in this application. You might use this source code as a guide to develop new function calls and messages. Send your discoveries to *Centura Pro* for all of us to enjoy! **CP**

Download WIN_API.ZIP from the* Centura Pro *Web site at www.ProPublishing.com, or from the Centura third-party forum on CompuServe.

Rajeev Mitra is a Senior System Officer in Citibank, NY. He can be reached at (914) 899-7585, rajeev.mitra@citicorp.com, or 103611.2506@compuserve.com.

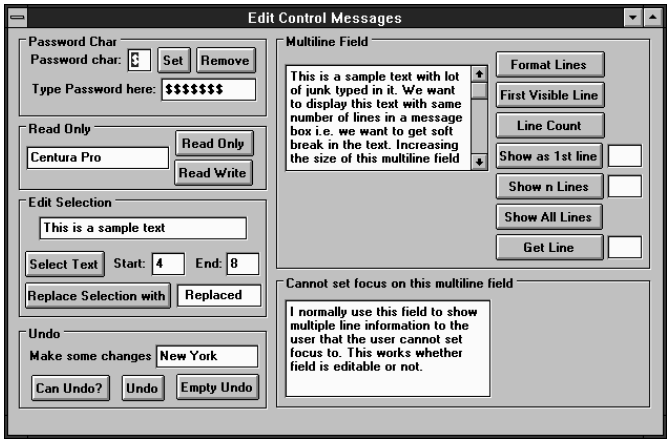

Figure 4. Fine-tune child windows with edit control messages.

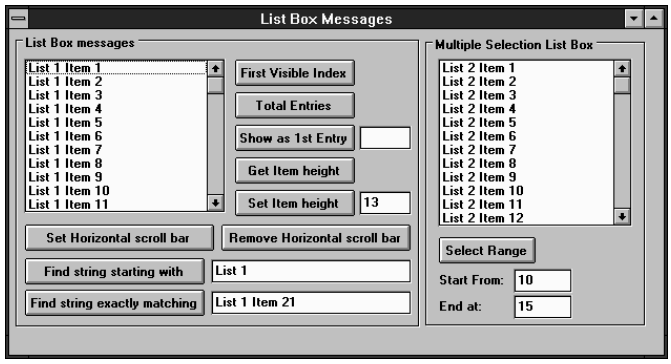

Figure 5. Control list boxes with greater precision.

Leap Onto the Web

Mark Hunter

ForeSite offers Centura the opportunity to run with a market-leading product in a hot new field: large-scale application deployment on intranets and the Web. Here's a brief technical overview.

InfoSpinner's ForeSite was designed to empower companies that need to service large numbers of clients and need to present dynamic data on the pages they return to those clients. Examples would include home banking, package tracking, merchandise catalogs, and similar applications where the data presented changes frequently.

Remember that ForeSite is not an HTML editor, nor an application development tool. ForeSite users would first build their Web pages using conventional editors. The resulting HTML is modified with PageManager, a ForeSite administrative tool. Additional special tags, like "<DYNATAG>" or "<DYNAANCHOR>", are placed in the Web pages. These act as signals to the ForeSite Adaptor, which will later trap such requests. These tags are assigned to specific Page Servers, which I define shortly. PageManager, then builds the Web site from the modified pages.

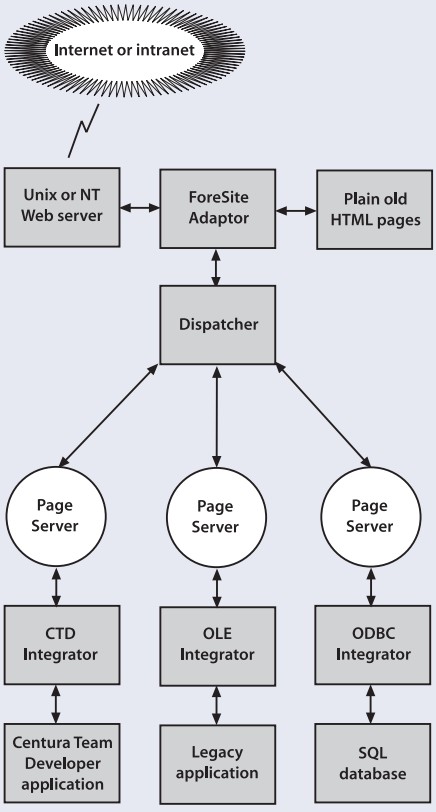

Figure 1. ForeSite traps URL requests for dynamic page content and routes them to individual Page Servers for fulfillment.

Figure 1 shows a simplified ForeSite architecture at runtime. ForeSite also has a set of tools for configuring the services attached to each Page Server, to configure the Dispatcher, to handle page caching, and to handle other administrative tasks. I omitted these tools—interesting in themselves—from the diagram for clarity.

To understand what makes ForeSite so powerful, you must learn a little more about each of the components in Figure 1. One of the most important concepts to understand is that all ForeSite components communicate with TCP/IP. This means that they have hardware independence. Only the Adaptor needs to be located on the Web server. Dispatchers, Page Servers, and the processes they link to may be located anywhere that works for the customer. A small site might have the Web server, Dispatcher, Page Servers, and linked processes all running on the same computer. A large site would have these spread out over many computers. In the descriptions that follow, remember that requests flow into these components and answers flow back through the same components.

Where the power resides

The "Adaptor" runs inside the process space of the Web server itself, providing especially fast performance. In contrast, typical CGI interfaces require spawning separate processes for each request. The Adaptor must determine whether the URL being requested is one that is administered by ForeSite. If so, the request is passed to the Dispatcher. If not, the request is passed to the Web

Continues on page 8

Got critical problems that require the services of an expert? Send them to Ace Centura, care of *Centura Pro* Editor, 71460.3142@compuserve.com. And now for this month's crisis . . .

Dear Ace,
I'm using SQLBase 6.0.1 for NetWare (3.12). I used SQLWindows (5.0.2) to write a complex application. I tested this application for half a year with five to 10 users. It worked well. But when I went into production (with more than 20 users), the application stopped working because of lots of locks. I got some tips from a Centura support technicians:

- To make a commit after all select and fetch statements (a commit after each insert or update statement I was already doing),
- To avoid sqlImmediate statements, and
- To create critical tables (tables with only a few columns and where lots of inserts are done by the users) with the parameter PCTFREE = 90

The application has now grown very slowly (because of lots of commits), but it works.

My problem: Next time the number of users will increase to more than 40. So I asked my Centura-friend, what to do. He recommended not to use SQLWindows (because SQLWindows only works properly with fewer than 30 users) and to use ORACLE.

Is this true? Many thanks for your help in advance,

Joerg Holl, BIT—Austria, holl@bit.ac.at

Dear Joerg,
It's difficult to address the problems you cite in your first application without more details. What isolation level are you using and what kinds of locking problems are you encountering?

By default, SQLWindows applications connect to SQLBase using the Read Repeatability (RR) isolation level. This isolation level provides good data consistency but impairs data concurrency, and is considered one of the "safer" isolation levels. If you don't change the isolation level from RR, then Shared locks will remain on the data for the duration of the transaction. When using RR, it is important to do a commit after fetching all the rows in a SELECT statement to release the shared locks. For this reason, it is also desirable to use TBL_FillAll with SalTblPopulate so as to not have to keep locks in the database any longer than necessary.

Generally, it is recommended that you use the Release Locks (RL) isolation level with SQLBase. This isolation level provides better data concurrency at the expense of data consistency. But by using another SQLBase feature, namely ROWIDs, you can manage data consistency as well. My approach is to start off with using the RL isolation level and then consider switching to another isolation level *if* necessary for the application being built.

With the RL isolation level, it is no longer necessary to commit after fetching the rows from a SELECT statement. In RL, the shared locks on the data are released almost immediately; the data is locked just long enough to build the result set.

Note that the isolation level has no effect on the duration of exclusive locks, which are established with an insert, update, or delete. A commit is always necessary to release an exclusive lock, regardless of the isolation level.

As for SqlImmediate, I agree with refraining from using it. Besides the history of bugs associated with SqlImmediate, there are two major problems in my opinion. First, you have no access to the SQL handle used with SqlImmediate, so you can't effectively do error handling when using SqlImmediate. Second, the implementation of SqlImmediate degrades performance. You are much better off connecting another SQL handle and using the standard functions SqlPrepare, SqlExecute (or SqlPrepareAndExecute), and SqlFetchXXX.

The third recommendation you cite from the Centura engineer can also be useful. Here the engineer is addressing deadlocks. Deadlocks occur when two or more users each have a lock on some data in the database, and they are each waiting to acquire a lock on data locked by someone else. Since SQLBase does locking at the page level, it's possible that two or more users are trying to lock the same page to insert a row into the same table.

One way to help avoid deadlock is to use the PCTFREE parameter when creating a table to minimize the number of rows stored per page, as suggested. However, additional disk space is used when this is done, as more pages are required to store a table.

Another tip for avoiding deadlock is to make sure that all tables in the database are accessed in the same order in all applications. This prevents user A using application A from getting a lock on data in table X and waiting for a lock on

Continues on next page

Continued from page 7

table Y, and user B using application B with a lock on table Y and waiting for a lock on table X.

Another area where deadlocks can be a problem is in indexes. For this reason, it's useful to limit indexes to those required for RI and those indexes that are used during queries.

Finally, in any multi-user environment, you can't avoid deadlocks avoided, so it's important that any production application in a such an environment be able to handle deadlocks in its SQL error handling.

For more information on these and other related topics I refer you to the official SQLBase documentation, especially the *Advanced Topics Guide*, and the SQLWindows documentation, specifically the chapters on interfacing with SQL databases. I also recommend that you read some articles that have appeared in earlier issues of *Centura Pro* (then known as *Gupta Pro*), specifically "On the Sunny Side: Optimistic Locking Strategies" from the May 1996 issue and "The Secret of Locking Around Data-on-Demand" in the June 1996 issue, both by Rick Greenwald.

With regards to your colleague's comment on using Oracle, it's important to compare apples to apples and oranges to oranges. SQLWindows is a tool for developing client-side applications, Oracle is a relational database server. If any comparisons are to be done, you'd want to compare SQLBase to Oracle and SQLWindows to Developer/2000 (Oracle Corp.'s high-end client-side development tool). In considering database scalability, SQLBase is certainly capable of handling 40 concurrent users. My personal experience with SQLBase is handling roughly 70 concurrent users, and I've seen reports of production databases with 120 concurrent users.
It all really comes down to database design and application code. SQLWindows can be used to build applications for multi-user environments for both SQLBase and Oracle. You just need to understand how to effectively and efficiently use the chosen database. Hope this helps.

Sincerely,

Ace

Leap Onto the Web . . .

Continued from page 6

server to be handled in the conventional way.

The "Dispatcher" is capable of using a "hot spare" concept for fault tolerance. It's responsible for determining which Page Server(s) can service a request, then passing that request on. In a high-volume environment, there would typically be many Page Servers servicing the same task. In this case, the Dispatcher also provides load-balancing functions. It accommodates security and can require a user ID and password to access specified Page Servers.

The "Page Server" is responsible for accepting requests and returning dynamic content back to the requester. Page Servers use Integrators to reach the actual process that will prepare the content. Page Servers can also cache pages. If a request can be satisfied by a page already in cache, response time will be exceptionally fast. Cache refresh can be time-driven or event-driven.

"Integrators" are the communication layer that link Page Servers to actual applications or databases. ForeSite initially had Integrators for any application that speaks OLE, or any ODBC database. Centura has developed another Integrator for Centura Team Developer applications running under Web Developer. It took a Centura engineer about two weeks to do this, so the level of effort involved is not too great. Since the ForeSite Integrator specifications will be widely published, any vendor can write an Integrator to reach their applications, and many are expected to do so. Imagine an Integrator for SAP R/3 or PeopleSoft applications. It's easy to see how quickly almost any application can be brought to the Net.

The laundry list

Considering how powerful and flexible each ForeSite component is, it's easy to believe Centura's claims. ForeSite promises to bring "Performance, scalability, reliability, extensibility, and security" to Web deployment, and it looks like they have the technology to back it up. Soon *Centura Pro* will feature actual customer experiences and an inside look at ForeSite training. Stay tuned! **CP**

Centura Code Warrior, Part 2

R.J. David Burke

In this exciting conclusion to last month's saga, our hero tackles test plans, testing, and debugging.

In the '80s we saw the shift from structured development and the waterfall model of the software development life cycle (SDLC) to rapid application development using an iterative approach. One of the most important things to come out of this shift was the realization that testing isn't a separate activity that occurs towards the end of the SDLC, but an integral part of all other development phases. Unfortunately, this was lost on the majority in the industry, in my opinion, and there is still little or no testing done during analysis and design phases. The problem, I believe, is one of education. This article hasn't the space to cover this topic in detail but here's a quick look at some important testing concepts.

Preparing to test

One of the most important documents you should develop before each testing session is a test plan (see a sample on *Centura Pro's* Web site). Formulating a test plan will help you understand specifically what you are testing, the scope of your testing, and so forth. I've seen various levels of formality in test plans. It often depends on the size of the organization, the experience of the developers, and the complexity of the project. What's important, of course, is that the test plan meet your needs and requirements. It should clearly identify the testing environment, test methods, criteria to be used, and, often, the expected results. A test plan should be traceable back to the original requirements of the system.

A typical test plan will include a description of the environments to test under: hardware, OS, connectivity, data to use, testing tools to use and so forth. This is followed by a list of scenarios to test and expected outcomes if doing regression tests. The scenarios expand upon the specific testing steps or activities to perform (often this can be automated with today's testing tools).

Types of testing

There are various perspectives on types of testing, all useful. The first take on types of testing addresses the scope or level of testing. This perspective comprises unit testing, integration testing, and acceptance testing.

Unit testing is the testing of individual software components, isolated from other software components. This type of testing is specific to a software component without regard to where it fits in with the rest of the system. Unit testing validates the technical and functional design integrity and is typically performed by the actual developers of software components.

With integration testing (also called system testing) various software components are tested in conjunction and cooperation with each other. Integration testing proves compatibility and completeness of the overall design. Depending on the number of components being integrated, this type of testing is typically conducted by a dedicated QA or testing department in larger organizations.

Finally, acceptance testing verifies the system against software requirements and specifications. Acceptance testing is usually performed by a mixed team from QA and from among the end users and typically measures performance and usability.

Generally, these types of testing are associated with testing code, but they apply equally to testing design specifications and analysis results as well.

Note that these forms of testing can be scaled as appropriate. For example, in developing a class, do unit testing for each class function, followed by integration on the class itself, and then acceptance testing by the class designer and the programmer who will use that class. The acceptance testing of the class is also the unit testing for using the class in an application, which is followed by integration testing of the application and ultimately, acceptance testing of the application.

Another take on testing

Another perspective on types of testing separates tests into black-box and white-box testing. This perspective on testing is orthogonal to the first perspective and can be used in conjunction with it.

White-box testing is concerned with testing the correctness of an implementation. In software, white-box testing involves testing all parts of your code. This typically means testing every If and Select Case construct,

the terminating conditions for While and Loop constructs, every menu item, every message, and so on. With white-box testing you "look" inside your components and make sure they operate correctly, completely, and soundly. Complete coverage of all code is an important goal for white-box testing.

Black-box testing concerns itself with testing the input to a component and getting the expected output. Black-box testing treats the component as a black box; you don't consider the implementation, merely the output with respect to input. Black-box testing is useful for testing conformance to requirements, regression testing, and validating the business use of a component.

The various combinations of testing types (unit, integration, acceptance, white box, black box) create an extensive test matrix that will give you a good idea of how much work needs to be done in testing and allow you to plan accordingly.

Incorporating the results

Once a testing session has been conducted, it's important to gather the results and present them in a useful way. Typically, this is done with a Test Analysis Results document that shows the outcomes of the testing scenarios. This information is the feedback that designers and developers need to continue to improve their deliverables.

Debugging code

Good coding practices and incorporation of test procedures will go a long way to bug prevention; but, with the current state of the art, you can still count on doing some debugging work. Fortunately, SQLWindows and CTD provide a fairly good debugging environment. The debug dialog and related dialogs are well documented in the product; no need to re-document them here. The animation facility can be useful in some situations, such as confirming execution of code paths, but I've found that I really don't use it that often. (But it does seem to make a good marketing demo for the product.)

In advanced applications you'll find situations where the debug can't be used effectively. This happens when you try to debug focus/activation messages or functions or when timers are involved. In such cases you may want write logging code in the application. Chapter 30 of *Special Edition, Using Gupta SQLWindows 5* (Que, ISBN 0-78970-189-8, published in 1995) discusses the undocumented SalLogXXX functions that can be useful in this regard.

Writing debug code

The usual approach to switching debug code on and off is to declare a global Boolean variable (for example, gbDebug) which is set to TRUE or FALSE, typically in the SAM_AppStartup message handler. Then you can put If gbDebug statements in your code as required.

When you're writing debug code, be careful that it's "read only" and doesn't alter the flow of execution. Otherwise, your ap

Listing 1. Debug code should be informative, but not alter non-debug behavio

```
If gbDebug
    If nCount < 0
        Call SalMessageBox( 'nCount is less than 0',
                '', MB_Ok )
If nCount < 0
    Set bRetVal = -1
Return bRetVal
```

Listing 2. Altering program behavior in debug code leads to even more proble

```
If gbDebug
    If nCount < 0
        Call SalMessageBox( 'nCount is less than 0',
                '', MB_Ok )
    Set bRetVal = -1
    Return bRetVal
```

Another useful tip is to avoid using nested functions in expressio
Consider:

```
Set nLength =
    SalStrLength( SalStrLeftX( SalStrScan( sSource,
    sTarget) ) )
```

At compile time, lines with erroneous nested expressions are harder to

```
Set nPos = SalStrScan( sSource, sTarget )
Set nSubString = SalStrLeftX( sTarget )
Set nLength = SalStrLength( sSubString )
```

When separated out, it's easier to see the problem at a glance, namely t
When stepping through code to debug it, it's much easier to follow

```
Call Foo( Bar(  ), Bar(  ) )
```

and you wonder if you're about to step into Foo or Bar as the first para

Visual Toolchest debugging support

The Visual Toolchest adds useful functions that support writing debug
VisDebugAssert provides capabilities in the spirit of the Assert ma
Assert macro is used to confirm assertions, conditions that you expect

Listing 3. Using VisDebugAssert.

```
Set hWnd = SalCreateWindow( frmX, hWndMDI )
If gbDebug
    Call VisDebugAssert( hWnd != hWndNULL, 'hWnd !=
        hWndNULL' )
```

VisDebugAssert is basically a useful shortcut for:

```
If NOT bExpression
    Call SalMessageBox( sAssertionMessage || ':FALSE',
                'Assertion Failure', MB_Ok )
```

Another useful debugging function from the Visual Toolchest is Vi
having the string delivered to a message box or to the Windows debug
Windows debug window. With DBWIN launched and DBF_MessageB
VisDebugString is that you can make the string output conditional dep
effectively use this set of functions.

Continues on page 12

Software Testing Resources

Mercury Interactive
www.merc-int.com

LoadRunner, a load testing solution, simulates virtual users to test the entire client/server enterprise (including: performance evaluation under varying conditions, and system tuning). WinRunner is an automated software testing tool for MS Windows-based client/server applications. TestDirector manages multiple phases of the testing process: planning and design, automated test creation, manual and automatic test execution, defect tracking, and application quality analysis.

Performance Awareness Corp.
www.PACorp.com

The company's latest version of winVue, release 2.0, is an automated testing tool for Windows. Test scripts are created by recording users' actions while they use the application. Organizes modular test scripts into sophisticated test suites. A suite manager allows testers to establish test script hierarchies and execute all or only a portion of the tests. A dedicated scripting language is provided for test customization. The debugger provides breakpoints, single-step execution, and the like, as well as information to determine why a particular test may have failed. Combined with the company's other products, pre-Veu-C/S and WinSatellite, this product supports single-user functional testing and multiuser stress load testing.

Platinum Technology
www.platinum.com

Final Exam is a suite of test tools for client/server application development. C/S-Test automates software testing cycles in three-tier client/server applications. Tests are executed remotely on multiple client workstations simultaneously and communicate with each other using both synchronous and asynchronous messaging. With C/S-Test, users can run tests based on familiar record and capture procedures or custom-program them using a C-like Test Management Language (TML) code; or they can combine recorded and user-programmed code in any script.

Pure Atria
www.pureatria.com

Last August Atria Software and Pure Software merged to form a new company. Atria offered software configuration management products and Pure developed developer diagnostic and application testing products. The newly-merged company recently announced intentions of acquiring Integrity QA, a venture-backed company founded by ex-Borland International staff, developing a next-generation Windows-based GUI testing product. That product hadn't been announced at press time. Pure Software's primary product was Purify, until recently a testing tool for UNIX applications. Prior to the merger, the company released a version for Windows NT. Its purpose is to pinpoint runtime errors and memory leaks throughout C and C++ applications, including ActiveX controls, OLE components, third-party DLLs and libraries.

Rational Software Corp. and SQA, Inc. (recently merged)
www.rational.com

Rational recently announced its intentions to merge with SQA, Inc. (www.sqa.com), makers of SQA Suite. SQA Suite consists of SQL Process, a formalized methodology for automated testing of Windows client/server applications; SQA Robot, a product that lets you create, modify, and run automated tests on Windows platforms; SQA Manager, which lets you plan, manage, and analyze client/server testing projects; and SQA LoadTest, SQA Process, which offers load, stress, and multi-user testing of Windows client/server applications on TCP/IP, IPX/SPX, or NetBIOS/NetBEUI networks. The SQA Test Repository is the foundation of SQA Suite. All products in SQA Suite are integrated with the SQA Test Repository.

The company also recently acquired Visual Test from Microsoft, a testing tool primarily useful on Windows applications developed in C and C++.

Segue Software
www.segue.com

Quality Works is an automated software testing solution that can validate all components of second-generation distributed client/server applications across multiple platforms and releases with one set of reusable scripts. It includes: QA Planner, a test planning and management module; QA Partner and QA Partner/Distributed testing systems; QA DBTester, a SQL-based database testing module; QA Partner Extension Kit, for handling custom objects; and QualityWare scripts, including GO! automatic testing and Audit! automatic GUI verification.

Vermont Creative Software
www.vtsoft.com

Vermont HighTest Plus is an automated software testing tool for Windows applications. It features recording that lets users translate all low-level mouse events to high-level object events that are completely independent of screen location and resolution. Its interactive Suite Manager combines scripts into test suites. Users can play a script or suite from the command line, play scripts and suites at a specific time of day, and read data from an application into script variables. A playback log file provides a report of the result of every command. **CP**

Listing 4. Making the most of VisDebugString and debug levels.

```
Global Declarations
    Constants
        System
        User
            Number: DEBUG_LEVEL_ALL = 0
            Number: DEBUG_LEVEL_WARNINGS = 1
            Number: DEBUG_LEVEL_ERRORS = 2
            Number: DEBUG_LEVEL_FATAL = 3
    .
    .
    .
    Call VisDebugSetFlags( DBF_MessageBox, TRUE )
    Call VisDebugSetLevel( DEBUG_LEVEL_ERRORS )
    Set gbDebug = TRUE
    .
    .
    .
    Set n = Foo(  )
    If gbDebug
        If n < 0
            Call VisDebugString( 'n < 0',
DEBUG_LEVEL_WARNINGS )
    Set m = Bar(  )
    If gbDebug
        If m < 0
            Call VisDebugString( 'm < 0',
DEBUG_LEVEL_FATAL )
```

Following the code in Listing 4, the application defines various debug levels. At some point during execution the debug level is set to DEBUG_LEVEL_ERRORS. Later, n is assigned the result of calling function Foo. Since we are in debug mode, n is tested to see if it is less than 0; but we are only concerned with this if the debug level is set to DEBUG_LEVEL_WARNINGS. In this case, no string is produced. Next, m is assigned the result from calling function Bar. If m is less than 0, it is considered a fatal error and the string is always produced.

I've yet to find an equivalent to DBWIN.EXE for Win32 for use with Centura Team Developer. If you are in the know, share your insights with me.

If you've developed a technique, strategy, or approach that improved your development productivity, drop me (and *Centura Pro's* editor) a note. I am currently working on a SAL coding guidelines document; your contributions are welcome. **CP**

R.J. David Burke is a Project Leader with Centura Software Corp. You can send him e-mail via David.Burke@CenturaSoft.com.

Auto-drop Combo Boxes — Centura Tip!

Sometimes it's desirable to have combo boxes drop down as soon as they receive focus. It's easy to do this using the Windows API. First, be sure that you have declared these global constants:

```
Number: WM_USER = 0x0400
Number: CB_SHOWDROPDOWN = WM_USER + 15
```

Next, you need to trap the SAM_SetFocus event for the combo box and send an API message at that time:

```
Combo Box: cbTest1
    Message Actions
        On SAM_SetFocus
            Call SalSendMsg( hWndItem,
                CB_SHOWDROPDOWN, TRUE, 0 )
```

That's all there is to it! For an example, see the source code file AUTODROP.ZIP.—***Rob Wierdsma***

Download AUTODROP.ZIP from the Centura third-party forum on CompuServe or visit the Centura Pro Web site at www.ProPublishing.com.

Case Study: Taking a SQLWindows Application to the Net

Susan R. Baker

The Internet has emerged as the "place to be" for new, customer-oriented applications. Many tools are available to help you build Web- or browser-based solutions. To integrate your existing SQLWindows application with the Internet, however, is a bit more complicated. In this article, the author discusses a three-tier application that uses the Internet as the communications layer while using SQLWindows as the front-end presentation layer. This combination of technologies and tools allows you to take full advantage of SQLWindows' GUI richness and perform extensive data checking and validation locally, and provide access (cheaply and easily) to corporate data.

The application used for this case study is INTELSAT's TVMax. INTELSAT (The International Telecommunications Satellite Organization) is the world's largest commercial satellite communications services provider with 24 satellites in orbit. INTELSAT, with its international customer-base, has traditionally conducted satellite capacity inquiries and booking processes with the use of telephones, faxes, and telexes. However, with TVMax, the door has now been opened to conduct these transactions over the Internet.

The TVMax application provides customers with remote-access communications to INTELSAT's Occasional-Use Television database. INTELSAT's Occasional Use service allows its customers to reserve occasional-use and special event television satellite time. Customers can query the current TV schedule, view the schedule graphically, submit requests for service, amendments, or cancellations, directly book certain services, and download the latest TV reference data to ensure accurate data entry. Also, customers receive instant acknowledgment of their service requests; in addition, for direct bookings, all parties involved receive confirmation messages.

TVMax V2 was originally released last year with a SQLWindows front end, an SQR middle-tier, and an Oracle back-end, using X.25 packet-switched networks. The SQLWindows client also includes SilverWare's Windows Communications Tool Kit to allow asynchronous modem communications. Requests for transactions are sent to a UNIX session at INTELSAT that invokes the SQR program. This SQR program verifies the request, performs the transaction, and sends back the resulting data, which the SQLWindows program then formats and presents it to the user. (See **Figure 1**.)

Enter the Internet

Following the implementation of TVMax V2, Internet capability became important. Although X.25 packet-switched networks are widely available (located in about 60 countries), the Internet is even more widespread (accessible in over 100 countries). Global access of this more modern technology prompted us to examine integrating TVMax with the Internet.

We considered the option of a zero-client, browser-based solution that eliminated software distribution. However, a browser- or Web-based solution would have required a complete rewrite of TVMax. More importantly, the SQLWindows presentation layer of TVMax performs a number of complex and necessary data checks, validation, and general integrity hand-holding. To incorporate these front-end features into a browser-based solution would require a language such as Java, which is not yet available in our current development environment. In part because of the imminent delivery date, we wished to keep as many of the tiers static as possible and just swap in a

different communications protocol, such as the Internet, as requirements dictated.

Our next step was to determine a strategy for using SQLWindows and the Internet. INTELSAT has an international and diverse customer base and many of them use 16-bit operating systems. At the time, Centura didn't have an Internet solution for that environment available. We then considered Spyglass. Spyglass's Client Web Technology Kit (WTK) provides the ability to add Internet connectivity to applications. The WTK provides a vast array of APIs and their associated functions to do practically any task imaginable on the Web or in Internet-related applications.

But Spyglass proved to be overkill for the TVMax application. Although the functions that TVMax needed were available in the WTK, they were a hard-to-find, not-well-documented subset of the overall package that came with a prohibitively high price tag.

Another alternative was to program TVMax using direct WINSOCK calls, as described in Mark Hunter's article in the October 1996 issue of *Centura Pro* ("Survive Windows Sockets"). This method required a more low-level programming effort than what we desired . Although we could have purchased a library or API that abstracted the WINSOCK calls for us, we would still have had to code certain necessary components (such as network layer issues and security features) into our application.

Those issues led us to consider Message-Oriented Middleware (MOM) as the solution to our Internet needs. MOMs, as with all middleware, sit between the application logic and the physical network. They insulate application developers from having to deal with the connectivity and network layers (such as TCP/IP and Sockets). MOMs are event driven in that, when an event occurs, the middleware application notifies a server that an action needs to be taken. They are generally asynchronous but can usually work synchronously if need be.

The chosen MOM

After looking at a few different MOM products (Iona's Orbix Talk, Momentum's XIPC, and Talarian's SmartSockets to name a few), we decided on TEMPEST Software's Tempest Messenger System (TMS). TMS is a new toolset based on open Internet/intranet standards that fits in extremely well with Centura products. (TEMPEST Software is the company that created the SQLWindows SAL-to-C Compiler.) It adheres to an e-mail paradigm, where "messages" (of any length or content) can be sent to a mailbox, broadcast, replied to, and forwarded. Many of the features that we would have had to code ourselves if we had decided on straight WINSOCK programming are built into the product, including:

- The need to choose and implement an open, proxyable protocol (TMS uses HTTP).
- Engineering high throughput and robustness into socket-listeners.
- Implementing and managing socket-based communications programs on several different platforms.
- Providing encryption and security models.
- Automatic load-balancing.

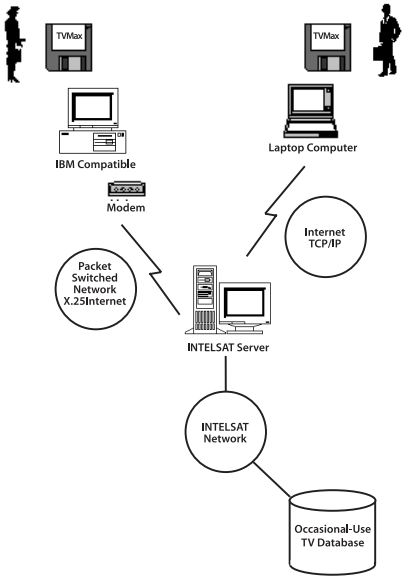

Figure 1. TVMax's architecture.

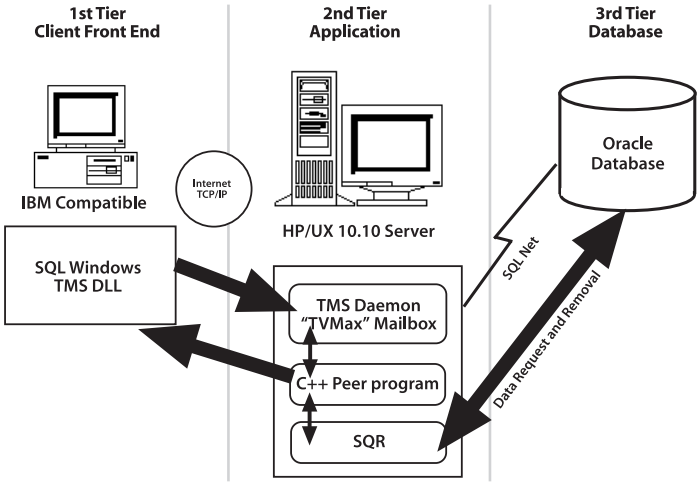

Figure 2. The three tiers of TVMax: the interface, application server, and database.

The API is remarkably simple to use. In TVMax's SQLWindows program, only a few lines of code had to be added to begin using TVMax on the Internet. The TVMax function that originally transmitted a request string to INTELSAT received the results (through X.25 and the SilverWare DLL) and then wrote them to a file. The revised function now transmits the request string via a TMS function call, receives the results, and writes them into the same file as the previous version. Because of the plug-and-play quality of the chosen middleware solution, no other code changes were needed on the client-side of TVMax to support Internet connectivity.

The "Peer" program, located on a UNIX server (written in C/C++, with some Pro-C thrown in for database access) also began quite simply. The Peer program acts as a daemon, polling the TVMax mailbox and handling the requests by spawning the SQR processes and then sending the output back to the client. This is done by stuffing the SQR results into a memory buffer and sending it by way of another TMS call. When the Peer program uses a Reply TMS call to send back the results, the message goes directly to the TVMax client that sent the original request. (It isn't rerouted back through the Mailbox from where the request was picked up.) SQR runs on the same machine as the peer program; however, the Oracle database it connects to can be on any machine as long as there's a way to get to that machine via SQLNet. The Peer program, itself, is quite simple. The most difficult part of integrating TMS with TVMax, on the server side, was the handling of administrative requests (such as starting and ending new Peer programs) and making the programs fault-tolerant. To make the TVMax system production level, it would have been necessary to program in these administrative features regardless of the chosen MOM.

Security

With TMS, there's no need for (and no way to get) the clients' or Peer program's IP addresses. All communications are handled through TMS' mailboxes. The only information that the interacting programs need to communicate with each other is the IP address and Port number of the TMS daemon's location. One feature that was a security requirement for any application accessing INTELSAT's corporate data over the Internet is the inclusion of Secure Sockets Layer (SSL). SSL provides server authentication, data encryption, and data integrity between a client and a server. It uses public key cryptographic technology from RSA Data Security, and SSL is the Internet standard. The beta version of TMS that we started out with lacked SSL. However, after hearing our SSL requirement, TEMPTEST Software began to incorporate it into the product (and should have released it by mid-February).

Ready for prime time . . .

With the expected delivery date of the latest version of TMS (with SSL included) in early February, we expect to go to beta testing mid- to late-February and production in April. The potential user base of TVMax could reach into the hundreds by the summer. Although TEMPEST's TMS is designed to enable very large-scale installations (thousands or tens of thousands), INTELSAT's current HP/UX servers (where TMS and Oracle reside) aren't

Companies Mentioned

Oracle
Oracle Corp., Redwood Shores, California. (415) 506-7000, fax (415) 506-7200, www.oracle.com.

OrbixTalk
IONA Technologies, Inc., Cambridge, Massachusetts. (617) 949 9000, fax (617) 949 9001, www.iona.com.

RSA Developer Toolkits
RSA Data Security, Inc., Redwood City, California. (415) 595-8782, fax (415) 595-1873, www.rsa.com.

SilverWare Windows Communications Tool Kit
SilverWare Inc., Dallas, Texas. (972) 247-0131, fax (972) 406-9999, http://rampages.onramp.net/~silver/.

SmartSockets
Talarian Corp., Mountain View, California. (415) 965-8050, fax (415) 965-9077, www.talarian.com.

SpyGlass
Spyglass, Inc., Naperville, Illinois. (630) 505-1010, fax (630) 505-4944, www.spyglass.com.

SQLWindows
Centura Software Corp., Menlo Park, California, (415) 321-9500, fax (415) 321-5471, www.centurasoft.com.

SQR Workbench for Windows
Sqribe Technologies (formerly MITI), Menlo Park, California. (800) 505-4399, (415) 614-0474 outside North America, (415) 326-5000, fax (415) 326-5100, www.sqribe.com.

TMS
TEMPEST Software, Inc., New York. (212) 984-1840, fax (212) 687-7609, www.tempestsoft.com.

XIPC
Momentum Software Corp., Hasbrouck Heights, New Jersey. (800) RUN-XIPC, (201) 288-5373, fax (201) 288-5474, www.momsoft.com.

quite so hardy. As more and more users begin using TVMax, we'll need to augment our current server side environment with more power to support the TVMax users as well as the INTELSAT users at headquarters that access the same data.

If we had it to do all over again

Although the Internet-enabled TVMax V3 is on track to be successful—bringing joy and business success to all who use it—if we were starting from scratch, we might do a couple of things differently. Currently, we are processing the TMS message calls, on the client-side, synchronously. Asynchronous messaging is supported (and preferred) by TMS, but using synchronous (blocking) calls worked like a charm and required the least impact to our existing TVMax code; so it was decided to stick with that method for now. Another potential change would be selecting a more appropriate language for our middle tier, currently written in SQR. SQR was originally selected because of in-house availability and expertise, and has proved to be a worthy component of our TVMax architecture. If TVMax usage grows to gargantuan proportions, however, a more robust language might be in order. All in all, TVMax V3 development and testing seem to be rolling along quite nicely, and demonstrates that putting a SQLWindows application on the Net can be done quickly, cheaply, and somewhat painlessly. **CP**

Susan R. Baker is a consultant for American Management Systems, Inc. She has been developing SQLWindows business systems applications for almost three years and can be contacted on the Internet at Susan_Baker@mail.amsinc.com.

Make SQLBase *Fly* on Windows NT

Centura Tip!

Do you run your SQLBase server engine on the Windows NT Server platform? If so, you should take a minute to make sure that the NT server itself is configured for optimum database engine performance. This tip should improve performance for other relational database servers, too.

NT is normally set up as a file server (the default settings). When it's set up this way, almost all of the server memory is cached for file activity, and the server doesn't want to give up any of that memory, especially for a mere application that's running. Unfortunately, the application in this case is the database server engine!

I discovered the following setting while testing out our NT server prior to converting from a Novell NetWare environment. I had been doing unload/load comparisons and was quite disappointed in the minimal performance gain of converting to NT. Once I changed this setting, however, NT absolutely flew! Queries that were previously taking eight to 10 seconds were returning in two to three seconds. A load of our major database went from 19 hours down to five hours!

Here's the way to change your NT server's setting to optimize your database server engine's performance:

1. Go into the Control Panel and select "Network."
2. Scroll down the "Installed Network Software" until you see the "Server" entry. Double-click on this entry.
3. You'll now be presented with the Server options screen. If you've got the "Maximize Throughput for File Sharing" radio button selected, change this to "Maximize Throughput for Network Applications," then click OK to save the changes. (See **Figure 1**.)
4. Click OK in the Network Settings and exit from the Control Panel.
5. Restart your NT server for the changes to take effect.

—***Mike Vandine*** **CP**

Michael is the Data Administrator for Argyle Diamonds in Perth, Western Australia. He has used Centura's products a long time and is a Certified SQLBase DBA. He's a member of TeamAssist and can be reached at 100240.1041@compuserve.com.

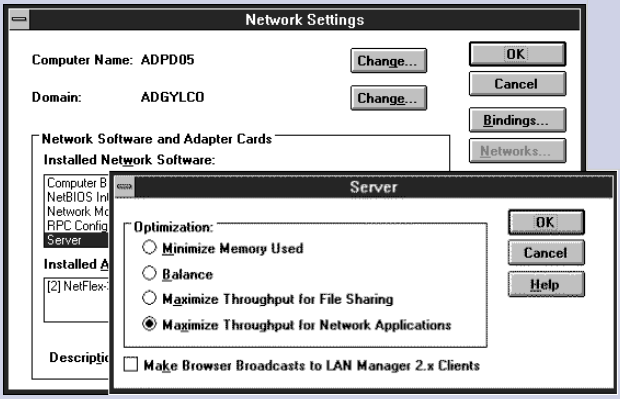

Figure 1. Windows NT database servers should optimize in favor of network applications.