Omni 7 *Directional*

OMNIS Marches Forward

This month represents the true re-birth of OMNIS as the company finally moves into marketing. Marketing, long forgotten as the means to sell product, has finally become the focus of OMNIS Software, Inc.

Several trade shows in the US, UK and Europe have resulted in the generation of substantial new leads, many of which will be converted into new OMNIS sales and therefore an expansion in developer numbers. The South East Asian region will also be a focus for marketing and several profile events have been planned for the second half of 1997.

The OSI share price has almost doubled this year as investors too return to the company. This is the true test of the durability of OMNIS and if the trend continues, as is expected, at last we will all have the profile we deserve!

DLA Announces Special Prices for South East Asian OMNIS Developers

As a result of DLA's close relationship with OMNIS Software, Inc, DLA is pleased to announce special OMNIS Studio Pricing effective for all orders received from 6 June 1997 until close of business at 5pm AEST Friday 29 August 1997.

OMNIS Studio is the next generation of OMNIS software and is a full Object Oriented development environment. OMNIS 7v3 will continue to be supported and developed for the foreseeable future thereby offering developers a choice between the current development methodology and the new Java/Active X world. OMNIS 7v1 continues to be a popular and robust development engine throughout the region. Richard Ure will report on its use in the next issue of **Omni**Directional .

DLA has negotiated these price discounts with OMNIS Software, Inc so that all South East Asian developers can commence using the latest version of OMNIS immediately and at the lowest prices in the world. No other jurisdiction is now offered these prices.

Product	Special Price	List Price
OMNIS Studio	A$1,749	A$2,299
OMNIS Studio Data Access Manager	A$1,449	A$1,899
OMNIS Studio Production Manager	A$ 849	A$1,099

Relevant conditions are as follows:
1.   OMNIS Studio is delivered with full hard copy User Manuals.
2.   Studio deployments accessing non-OMNIS SQL backends are free!
3.   Native Data File (now called 'Cache') will be available at significant discounts for volume. Developers will be able to 'lock in' at a pre-determined quantity for the year and buy at a discount price. If the quantity is NOT met at the end of the year, an adjustment in price per unit will be made.
4.   Maintenance is *very* important for these tools. Four (4) updates per year are scheduled and committed to by OMNIS Software and therefore it is really a requirement that the software be the subject of a Maintenance Agreement.

Full details of theses software components are contained in **Omni**Directional #9 a copy of which is available in Acrobat format on The DLA Web site at <http://www.dlagroup.com.au/omnidirectional>.

Orders for OMNIS products may now be completed via The DLA Web page. Please otherwise contact Amanda Turner <amanda_turner@dlagroup.com.au> at DLA to place your orders.

David P. Lewis
CEO
The DLA Group
david_lewis@dlagroup.com.au

Loading Selection Lists

A common interface feature in database applications is to offer the operator a list of choices as a means of selecting options, parameters or parent records. It can also be used as a navigation technique. The operator double-clicks on their choice from this list or performs some other action after selecting a line to indicate that the selection has been made. Popup and drop down lists are also used for this purpose.

Often these lists represent records in the active datafile, but sometimes they are created simply for the convenience a selection list offers. In the latter case, a database table may be created to make building this list more efficient, to allow for additions to the list without access to the underlying code, or to serve as a translation table for stored coded values.

For purposes of this discussion, there are two broad categories of information contained in such lists: *dynamic* data that can change at a moments notice based on input from any user of the system and *static* data that rarely, if ever, changes, and then is only changed by a system administrator. It is this static data that I address in this article.

Let me clarify with some examples. Dynamic data could be such things as a list of invoices for a specific customer or some other type of transaction. Static data might be the acceptable payment or shipping methods allowed on such invoices, or a list of "canned" messages that can be included in the message field of an invoice without typing. Sometimes, as in this brief example, these lists can be very short. Radio buttons may be used instead of lists in such cases if the cost in screen real estate is not too great. But static data can also include things like a list of the products offered for sale. These may change over time, but only at specific and well-defined points in time. Such a list may be a few hundred, or a few thousand, lines long and not suitable for selection using radio buttons.

The Basic Problem

Static selection lists are most often loaded when a library is first opened and maintained in memory variables. Since it is assumed that the data they contain will not change during a session, there is no need to be concerned about refreshing these lists. Lists of dynamic data, on the other hand, should be loaded each time they are to be used to be certain that the data they contain is current (at least to the point in time when they are loaded). Loading multiple selection lists over a network, either from a native OMNIS data cache or from a SQL host, can take a long time — especially if some of those lists are large. Avoiding a one to five minute delay each time such lists are needed is the very reason they are loaded at startup and held in RAM, but a long wait time at startup is still undesirable.

The Basic Solution

What if we were to load the lists already built from a File where we maintain them in the database itself? Instead of gathering a few hundred record images, OMNIS only has to retrieve a single field value (the list itself) for each selection list. The records represented in the list still exist in the database, so dependent records can be connected or related to them, but we don't have to read each record in its entirety (native OMNIS) or cause our server to build a select table (SQL) each time another workstation opens our library. The transfer to our RAM-based lists occurs much faster.

Yes, I know. This breaks one of the fundamental rules of database design — that of not storing the same data redundantly. But a more practical rule that carries much more weight with clients is to not waste their operators' time. In the world of programming for hire, academic rules lose out to practical ones quite often. The *reasons* for the academic rules are still valid — in this case, the fear of the list and the data becoming out of synch — but these needs can be satisfied using a well-designed workaround. The technique I will show you here can significantly reduce operator idle time on startup, even if it does take a little more coding work on our part to implement.

The List File

I recommend using a File Format that contains a field for each selection list we need

to establish. In use, this File will only contain one record and will work in a similar way to the System Constants record I first described in my first book on OMNIS (*Unlocking OMNIS 3 Plus*) years ago. These lists should *not* be kept in the System File, however, unless they are very small and very few in number. A record containing a number of large lists is a lot of baggage to carry around in the CRB in a record that gets updated often (requiring rereads in preparation for each update). A separate List File works best.

An alternate method is to store only one list per record in the List File, but then it is necessary to also store the name of the memory field the record is to update. In use, this method is significantly slower than the single record method since OMNIS has to retrieve multiple records. The whole idea of this method is to minimize the conversation between workstations and the server required by multiple record reads.

If your application is in SQL and your platform of choice does not support storage constructs that directly translate to OMNIS lists, you may find it beneficial to use a shared native OMNIS datafile for this purpose. If your selection lists are *truly* static (that is, the data *never* changes) and the price of OMNIS runtimes seems too much to bear (or is not justified by this timesaving technique alone), you could try deploying a copy of the datafile containing the list record to each machine and have it used locally (unshared). In fact, the library file itself can be used as the datafile for this purpose, but that's another article…

The List File is defined with a field for every list to be stored and either a Sequence field or a uniquely indexed ID field (primary key), since we need some means of retrieving the record during our startup process. The default file mode is set to *closed*. During startup we will set the List File to read only (reversibly, if possible), locate the record and transfer the lists using a series of Calculate commands. Calculating one list as another transfers the *entire definition and contents* of the source list to the target list, *including* all list statistics values (#L, etc.) and the foreground and background selection states of the lines of the source list. (For more detailed information on list procedure commands, see OMNIScience articles on list operations in past issues.) A subroutine called for this purpose would look like this:

```
Begin reversible block
        Set read-only files {List File}
        Set main file {List File}
        Single file find on LI_SEQNO (exact match) {1}
        If not(LI_SEQNO)
                OK message {List record does not exist.}
                Quit procedure
        End if
End reversible block
Calculate PAY_TYPE_LIST as LI_PAY_TYPE_LIST
Calculate SHIP_TYPE_LIST as LI_SHIP_TYPE_LIST
Calculate MESSAGE_LIST as LI_MESSAGE_LIST
; more Calculates as required
```

Notice that we must set the file mode *before* setting the List File as the Main File because a closed File cannot be made the Main File. I only set the Main File in this subroutine because I may not have set a Main File in the main startup routine at this point and a Main File is required in order to perform record location commands (Find, Next, Single file find, etc.). Single file find does not require that the record

retrieved belong to the Main File, but a Main File must still have been declared.

Establishing The List Record
To use the List File record, we must first create it. This is a onetime process that you would perform on the client's datafile before deploying the library that implements this list loading technique.

The procedure is quite simple. We build all the selection lists from their respective Files, transfer those lists to the fields of our List File, and then create the record. We need to make sure that a record does not already exist in the List File, since more than one record would cause some confusion. Here is what our procedure (method) would look like. There is no difference between OMNIS 7 and OMNIS Studio.

```
;build selection lists from Files
Begin reversible block
        Set read/write files {List File}
        Set main file {List File}
        If sys(83) ;; test if a record already exists
                OK message {List File already has a
record.}
                Quit procedure
        End if
End reversible block
Calculate LI_PAY_TYPE_LIST as PAY_TYPE_LIST
Calculate LI_SHIP_TYPE_LIST as SHIP_TYPE_LIST
Calculate LI_MESSAGE_LIST as MESSAGE_LIST
; more calculates as required
Prepare for insert with current values
Update files
```

List Definitions
By the way, it might be useful to discuss just what should be included in these lists. Remember, we want to keep our RAM footprint as small as possible while offering the most flexible interface. If the File associated with a list is made up of only a primary key field (code), a sequence number, and a name or description for use in the list, the list can be defined as the entire File using the File name in the *Define list* command. (Good arguments can be made that the primary key *or* the RSN are required, but not both, since one method of uniquely locating a record is sufficient.) It is likely that only the name/description will be displayed in your selection list field. If the name/description is indexed (and hopefully unique), only that one field is really needed. But beware — the combination of a 20 or 30 character field and associated index requires more disk storage than that field unindexed with a two to four character or a numeric (long integer) primary key!

If the list is used to represent a File with a more opulent collection of fields, such as a Product File, only the fields to be displayed for selection and a primary key or sequence number are needed in the list definition — certainly *not* the entire File definition!

Maintenance Problems
Some lists that will benefit from this technique may still require occasional maintenance. Even static information can change as business practices or other needs evolve. The list of canned messages is a good example: standard invoice messages come and go with each sales campaign and pending price increase! To successfully use this technique, we must also build maintenance facilities that allow

our selection lists to be updated automatically as the data on which they are based changes.

For best results in a large multiuser installation, these operations should be performed during a regular maintenance cycle when all users but the system administrator are logged off. Again, this technique is designed for background data that rarely changes, not transactional data that is constantly in flux.

There are three basic changes that we must be concerned with: inserts, edits, and deletions of records that correspond to lines in our selection lists. In all three cases, the list in the List File record must be replaced once the appropriate action has been taken on the list used in RAM. We only have to replace the list that has been changed, not every list in the List File record.

The procedure for updating the List File record is simple and to the point. No matter what change has taken place, we only need to replace the list with its most recent version. (Again, this process assumes that, in a multiuser installation, only a system administrator is performing this update, and that the update is done during a maintenance cycle, not by any and all users during the normal work day.) This process would be called as a subroutine from each of the procedures (insert, edit and delete) that could change the current contents of the list. Here is that procedure:

```
Begin reversible block
  Set read/write files {List file}
  Set main file {List file}
End reversible block
Single file find on LI_SEQNO (Exact match) {1}
Prepare for edit
Calculate LI_MESSAGE_LIST as MESSAGE_LIST
Update files
```

The means we use to get to the point where this subroutine is needed is a bit more elaborate. In my work I create a *maintenance window* for each selection list and its corresponding File. An example of such a window is illustrated here.

Maintenance Windows

A list maintenance window contains a list field for the list being maintained and fields for data entry of whatever data is held in the corresponding File. This window will only be dealing with its corresponding list and the File that goes with it, which will be the Main File while this window is open. When we open the window, we would like to see the data fields represent the record whose image is selected in the list if a line is currently selected (or cleared otherwise). Also, we will be sorting our list so that its contents are in some order that will make sense to our operators, so we can set up our sort criteria (reversibly) upon opening the window as well. Our window initialization procedure ($construct method) will then look like this:

```
Begin reversible block
        Set main file {Message file}
        Set current list MESSAGE_LIST
        Clear sort fields
        Set sort field {MS_TEXT}
End reversible block
If #L
        Single file find on MS_SEQNO (Exact match)
{lst(MS_SEQNO)}
        Redraw named fields MS_ID_NUMBER to MS_TEXT
Else
```

```
        Clear selected files {Message file}
End If
```

Clicking on a line of the list should retrieve and display the corresponding record from the File emulated by our list. This is assured by the following procedure for the list field. Here is the OMNIS 7 version.

```
If #CLICK
        Single file find on MS_SEQNO (Exact match)
{lst(MS_SEQNO)}
        Redraw named fields MS_ID_NUMBER to MS_TEXT
End If
```

In OMNIS Studio, this would go in the $event method of the list field and would look slightly different:

```
On evClick
        Single file find on MS_SEQNO (Exact match)
{lst(MS_SEQNO)}
        Redraw MS_ID_NUMBER,MS_TEXT
```

Our other basic navigation commands also operate on the list rather than directly on an index in that File. Next and Previous can be reconfigured to traverse the list. Find is replaced by list field navigation. The contents of the list should be maintained in some rational sort order (usually alphabetical) so that our Next and Previous make sense to the operator. The Next procedure might look like this (Previous will be similar):

```
If #L<#LN
        Calculate #L as #L+1
        Single file find on MS_SEQNO (Exact match)
{lst(MS_SEQNO)}
        Redraw windows ;;OMNIS 7
        or
        Redraw windowname ;;OMNIS Studio
Else
        Sound bell
End If
```

Insert and Edit commands must disallow the list from becoming the current field during data entry, since this could confuse the operator. (We also don't want a different record selected using the list while we are editing the current one or entering a new one. We can manage this using a window control procedure in OMNIS 7 as follows (EDIT and INSERT are Boolean variables in a Memory-only File and are reversibly set to kTrue at the beginning of the Edit and Insert procedures respectively as we will see below):

```
If #EF=1&(EDIT|INSERT)
        SNA set current field {2}
        Quit to enter data
End If
```

Insert

The insert command looks like any other insert command with a few additional features. First, we have to manage the list that is displayed on the window, then we have to update the corresponding list in our List File with any changes.

Managing the display list has three steps. First, we must remember what line was current before the insert began so we can recover the view the operator had in case they cancel the process. Second, we need to deselect the current line since our prospective record does not yet have a line of its own.

Finally, we need to insert the new record into its properly sorted position in the list. The following code segment does these things admirably — with a few additional nuances.

```
Begin reversible block
        Set read/write files {Message file,System file}
        Set window control procedure 499 {Add control
procedure}
        Calculate INSERT as 1
        Calculate #1DO as #L
End reversible block
Clear main file
Calculate #L as 0
Redraw windows
Enter data
If flag false
        OK message (High position) {Record addition
cancelled at your request.}
        Calculate #L as #1
        Single file find on MS_SEQNO (Exact match)
{lst(MS_SEQNO)}
        Redraw windows
        Quit procedure
End If
Single file find on SY_SEQNO (Exact match) {1}
Prepare for insert with current values
Calculate SY_NO_MESSAGES as SY_NO_MESSAGES+1
Calculate MS_ID_NUMBER as jst(SY_NO_MESSAGES,'-
3PO')
Update files
Add line to list
Sort list
Call procedure (Do code method) Update list
Redraw windows
Call procedure (Do code method) Replace list record
```

The *Update list* procedure called at the end of the *Insert* procedure is used to locate the proper line in the list after the sorting has taken place. Remember, when you sort a list, #L does not change, but the *contents* of the current line very well might! The following subroutine, *Update list*, will make everything right again.

```
Begin reversible block
        Calculate #5 as MS_SEQNO
End reversible block
Set search as calculation {MS_SEQNO=#5}
Search list (From start)
Clear search format
```

Edit
The Edit command for this window has the same basic additions to the core edit process as our Insert command above. It is given here for the sake of completeness. Notice that we still sort the list because the displayed text for the edited line may have changed.

```
Begin reversible block
        Set read/write files {Message file}
        Set window control procedure 498 {Change
control procedure}
        Calculate EDIT as 1
        Calculate #1DO as MS_SEQNO
End reversible block
Prepare for edit
Redraw windows
Enter data
If flag false
        Cancel prepare for update
        OK message (High position) {Change cancelled at
your request.}
        Single file find on MS_SEQNO (Exact match)
{lst(MS_SEQNO)}
        Redraw windows
```

```
        Quit procedure
End If
Update files
Replace line in list
Sort list
Call procedure (Do code method) Update list
Redraw windows
Call procedure (Do code method) Replace list record
```

Delete
The Delete command must delete both the record in the File and its corresponding line in the list. Furthermore, it should retrieve the record corresponding to the line of the list that becomes current once the current line is deleted since our bookmark in the list (#L) should remain where it is. No sorting is required because the remaining lines will still be in their proper order. Here is a simple Delete procedure.

```
Begin reversible block
        Set read/write files {Message file}
End reversible block
Delete with confirmation {Are you sure you want to delete
this message?}
If flag true
        Delete line in list
        Call procedure (Do code method) Replace list
record
        If #L
                Single file find on MS_SEQNO (Exact match)
{lst(MS_SEQNO)}
        End If
        Redraw windows/windowname
End if
```

In the case where records from other Files might be connected to the one we propose to delete, a check should be made at the beginning of the procedure that there will be no orphaned children caused by the deletion. Only unencumbered records should be allowed to be deleted.

Further Considerations
We must make certain that no line is selected in each of these lists (unless there is a requirement that a *specific* line is preselected in a specific list). When the lists are freshly built, this is not an issue since #L is initialized to 0 by the *Define list* command and not altered by the *Build list* command. But when we are updating a list that has been in use, we should set #L back to 0 for that list so it is loaded with no line selected, or set to a specific default if that is required.

Seeing Is Believing
If you do your development in a single user environment (with your data residing on a local hard drive), but deploy to a multiuser environment, you probably don't notice the slowdown your users must endure when loading lists such as these. Here is a useful tip for testing data access speed when you are trying to determine whether one method is more efficient than another. Perform your speed tests with your test data on a local *floppy* drive. This emulates access through heavy network traffic well enough for you to see advantages of one technique over another.

If you need a more extensive set of example code using this technique, I will be posting a couple of example libraries on my web site (hopefully before this issue hits the streets). There will be one in OMNIS 7 and one in OMNIS Studio. Just follow the links from the home page at http://www.polymath-bus-sys.com.

What About Dynamic Lists?

It is conceivable that a similar technique could be used to manage selection lists of dynamic data and make them better candidates for RAM storage. What is needed is some trigger device that will cause all workstations with our library currently open to reload their copy of a selection list when it is needed if it has been updated. That is a topic for another time.
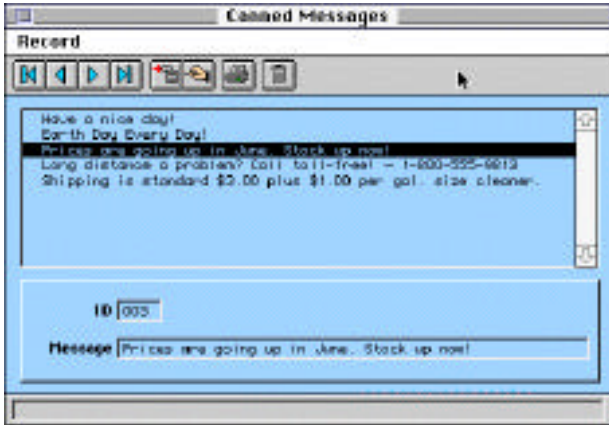
David Swain, owner of Polymath Business Systems in Bedford, New Hampshire has been teaching OMNIS to eager students around the US since early 1985. His technical journal, OMNIScience, remains a great source of detailed information on important features of OMNIS. You can learn more about these offerings through his Web site at http://www.polymath-bus-sys.com.

Stephen Miller

Stephen Miller is DLA's OMNIS Business Division Technical Sales and Support Manager. Stephen is responsible for DLA's technical support, training and consultancy services.

Stephen manages DLA's software development team and co-ordinates the OMNIS consulting division. He is responsible for all major development projects throughout Australia.

If you are wondering why he is smiling, have a look at http://www.dlagroup.com.au/staff/Stephen.html

OMNIS software launches nine-city European tour highlighting new products and executive management

President and CEO Tim Negris Addresses International Distributors and VARs

SAN BRUNO, CA, June 17, 1997 OMNIS Software, Inc. (NASDAQ: BLYH) today announced that it will begin a nine-city European tour to launch OMNIS Studio, the first Rapid Application Development (RAD) tool to deliver true component engineering software that is cross-platform, cross-database, cross-object (Java and ActiveX) and cross-architecture.

The tour will also highlight new management appointments within the company. President and CEO, Tim Negris, vice president of global sales, Peter Mork, and vice president of marketing, Pat McEntee, will meet with international press and analysts, customers, OMNIS distributors and OMNIS VARs throughout Stockholm, Munich, Dusseldorf, Paris, London, Birmingham, Brussels, Amsterdam and Milan.

In May, a successful reinvention of the company and product launch in North America at DB Expo generated over two thousand new customer leads for OMNIS Studio, strong recognition and support from the financial investment community and an explosion of Web site hits to the company's home page. Building on this momentum, OMNIS Software, Inc. is taking its corporate messages, products and newly-appointed executives to Europe for what promises to be one of the most aggressive marketing marathons undertaken by the company.

"With over half of our customers headquartered internationally, we feel it is vital to join force with VARs and distributors and directly focus on OMNIS Software's recent growth, new management and new products," said Negris. "Over the past several months, OMNIS has reemerged as a major player in the RAD tools and component engineering market."

OMNIS Software, Inc. international customers, VARs and distributors include: Key Solutions, Access Accounting and Eleetix Software in the United Kingdom, Aware and Equation in France, Software Products Italia and Q-Soft in Italy and Ericsson and Radisoft AB located in Scandinavia. In South East Asia The DLA Group are spearheading the spread of OMNIS.

OMNIS Studio is the first product in the marketplace which allows developers and companies to easily build applications by:

> Incorporating JavaBeans, ActiveX and custom components in the same environment
> Immediately deploying a language tool set for representing relationships between information on a Web site, in a database and within other applications.

General availability of OMNIS Studio, OMNIS Studio Production Manager and OMNIS Studio Data Access Manager was announced on May 5 at DB Expo in the United States.

Converting from OMNIS 7 to OMNIS Studio

Converting library files

Before proceeding make a backup copy of your library!!!!

OMNIS 7v2 and earlier
If you would like to convert a library created in OMNIS 7v2 or earlier you must first convert the application to OMNIS 7v3. There is a demonstration copy of OMNIS 7v3 located within the convert folder in your OMNIS folder. You can use application to convert your library to OMNIS 7v3.

OMNIS 7v3.x Pre-conversion tips
1. OMNIS Studio only supports icons up to 48x48 in size. If you have any icons larger than this in your OMNIS 7v3.x library they will not convert correctly.
2. Remove all trailing spaces, spaces and non-alphanumeric characters from format names. OMNIS Studio does not allow trailing spaces in class names and notation containing such names will fail.
3. The following will cause the call method command, formerly the call procedure command, to fail once the library is converted:

    Where a method in a different library is called

    Indirection is used

    Ambiguous names are involved (use unique names)

    An old window procedure has no name and was called by number alone

    Numeric or a name containing special characters is used (ie. - . /)

4. Use find and replace, found in the edit menu, to find every occurrence of $cwind and replace it with $cinst.
5. Find and replace every occurrence of $cclass and replace it with $cinst.
6. Remove any code behind extended fields and place it on a new procedure line within the same format. This will prevent the code from being removed during conversion.

Converting OMNIS libraries
Warning: You cannot convert a library that is locked. Doing so will render the library useless.

1. Make a backup copy of your v3 library.
2. Start OMNIS Studio and select the browser from the view menu. This will open the library browser.
3. Select open from the library menu in the library browser menu bar and hold down the alt key (option key on the mac), while pressing the open key, to open your library. This is done to stop the startup task from executing once the library is converted and prevent the possible loss of the error log. You may receive the following message:

Windows: "Path\application name needs to be converted. Afterwards it cannot be used with an earlier OMNIS. Convert library?" Yes/ no

Mac: "Path: Application_name needs to be converted. Afterwards it cannot be used with an earlier OMNIS. Convert library?" Yes/no

Warning: Your library will not convert properly if you try to open it by dropping it on the OMNIS Studio icon!)

4. Release the alt key (option key on the Mac) click the 'yes' button. You will receive the following message:

"Make sure there is plenty of disk space and only continue if you have a secure back up. Convert library?

Note: Your converted library may be up to 60% larger.

5. If these two conditions are true click the 'yes' button. The conversion might take a while.

Note: During the conversion process any errors that the converter finds will be written to the trace log.

Converting data files
OMNIS will automatically try to convert your 7v3 data file when you open the data file or your converted application attempts to open an old data file.

1. Backup your data file.
2. Select data file browser item from the view menu.
3. Select open from the data file menu in the data file browser window and open your data file.
4. You will receive the following message:

"Path\data file name needs to be converted. Afterwards it cannot be used with an earlier OMNIS. Convert data file?" Yes/ no

5. Click the 'yes' button. A very large data file may take several minutes to convert and may require reorganizing.

Converting icons
If you have made any changes to your OMNIS 7v3.x version of the omnispic.df1 you need to convert it. Otherwise, skip this section.
1. Make a backup of your omnispic.df1
2. Quit OMNIS Studio
3. Remove or rename the userpic.df1 file supplied with OMNIS Studio in the icons folder
4. Restart OMNIS Studio. You will receive the following message.

"There was a problem opening the icon file path\userpic.df1 maybe it needsConverting to the new format using the icon editor."
Click on the ok button.

5. Select the icon editor option from the tools menu
6. Select the convert old omnispic option from the file menu in the icon editor menu bar
7. Find and select your old omnispic.df1 file. The following message will appear:

"Path\omnispic.df1 needs to be converted. Afterwards it cannot be used with an Earlier OMNIS. Convert data file?" Yes/no
Click the yes button.

8. You will be prompted for the name and destination of the converted file. Go to the icons folder in the main OMNIS folder, where OMNIS Studio is located, and save the converted file as userpic.df1. This will take approximately a minute or two.

9. Restart OMNIS Studio

When you restart OMNIS Studio the userpic.df1 file will be automatically loaded. Any new icons you plan on adding to your library should be placed in the userpic.df1 file.

Note: Following conversion some objects in your library may have the wrong icons since the omnispic data file gets priority over userpic in id conflicts.

Graphs

The converter cannot convert the graphs in your library, therefore you will have to replace them with graph external components available in the component store.

Post conversion tips

It is strongly recommended that you read the conversion manual. These conversion tips are not meant as a replacement to that document. There are many areas that are not covered in this document:

1. Search your library for the word 'converter' to find the areas of your code that were altered during the conversion. Make sure the code still functions the way did prior to conversion.

2. Use the method checker, located in the tools menu, to check for errors.

3. Replace any extended areas in your library with the proper external components (ie. Replace graphs with the graph external component available in the component store)

4. Search your library for x9 to find out which external commands calls did not covert correctly. Check your original 7v3.x library to find out which command your OMNIS Studio code should call.

5. If you are using multiple libraries the close library command now close all libraries. To avoid this you must specify the name of the library that you would like to close.

6. The converter removes each toolgroup from window or menu formats and converts them to individual toolbar classes called t_oldwindowname or t_oldmenuname where old..name is the name of the converted window or menu format.

7. Toolbar system is totally different in OMNIS Studio. Calculating $dockingareas… is no longer the way to add docking areas to a window. You must now assign $toolbarpos in order to define a docking area.

8. If you created your own tab panes in OMNIS 7v3.5 they will not convert correctly and need to be written using the tab panes provided in OMNIS Studio.

9. Help messages behind window fields are transferred to the tooltip property for each window object and messages contained in menu formats are converted to the helptext property located in the property manager

10. Use find and replace utility to search on $was in order to find all notation that uses notation that is now obsolete.

11. In order for 7v3.x database events, such as #next and #previous to work correctly the $v3events preference needs to be turned on.

12. On the mac fields without borders now take the default background window color which is now gray. This can reset.

13. Popup menus implemented in 7v3.5 no longer work in OMNIS Studio. Context menus should be used instead to implement menus that popup over windows or window objects.

14. #Colors system table and all associated notation has been removed. Colors are now represented using rgb values. This may cause all code in your converted library that tries to set the color index of an object to fail, or color some objects will be different than expected. Color constants should be used instead.

15. Use find and replace to locate the following commands. If the command is marked as an obsolete command then it is ok and will function correctly. If the command references the fields by number, change it to reference the fields by name. Failure to do so will cause your library to crash.

> Disable fields
> Enable fields
> Hide fields
> Queue click
> Queue double-click
> Queue scroll
> Queue set current field
> Send to a window field
> Show fields

16. All dams have been renamed in OMNIS Studio; their names now begin with the letter "d". For example, the Oracle DAM is now called doracle.dll under windows, or doracle on the Mac.

17. All externals, including extended command and function packages, have been renamed also in OMNIS Studio; their names now begin with the letter "x". OMNIS Studio loads certain externals automatically so you don't need to make any changes, but you may need to change any references to externals in your own libraries. If an external is missing or cannot be found, references to it will cause a runtime error.

18. Tables have now been renamed complex grids. Most of the functionality has remained the same but they should be tested thoroughly in order to ensure that they behave as they did in OMNIS 7v3.

Compiled by:
Bob Jensen
OMNIS Software

All New DLA Web Site

DLA are proud to announce a major update of our web site. The address remains the same, http://www.dlagroup.com.au but the site has a completely new layout. It contains a wealth of information about new OMNIS Software developments, as well as older products which are still in widespread use. As well, visitors can browse the collection of useful OMNIS resources and order OMNIS products *online!* If that isn't enough, you can also have a look through the mug-shots gallery and see what the DLA staff *really* look like! See you soon at our home on the Web.

Win32 API Calls

I have finished a new version of my API example library and loaded it on to the DLA Web/FTP Site in the general section as API32.zip.

Where RegDll is noted as the external call as opposed to RegisterDll this indicates I have used an external from Lars Toernqvist <100414.2146@compuserve.com> which is shareware. The use of this external was necessary because RegisterDll can be unreliable with Win32s. (RegisterDll and CallDll are reliable with Win16 calls).

The Win16 code is embedded in this application and the correct menus will become available for the OS you have utilised.

Some basic calls included are as follows:

Set Current Directory to Temp

Calculate #S1 as
REGISTERDLL('KERNEL32','SetCurrentDirectoryA','JC')
CALLDLL
('KERNEL32','SetCurrentDirectoryA','C:\Win95\Temp\')
with return value #2

Current Directory

Calculate #S1 as
REGISTERDLL('KERNEL32','GetCurrentDirectoryA','JJC')
CALLDLL ('KERNEL32','GetCurrentDirectoryA',255,#S1)
with return value #1
OK message {Current Directory is [#S1]}

Beep

REGISTERDLL('USER32','MessageBeep','V')
CALLDLL ('USER32','MessageBeep')

Get Foreground Window

REGISTERDLL('USER32','GetForegroundWindow','JV')
CALLDLL ('USER32','GetForegroundWindow','') with
return value #1
REGISTERDLL('USER32','GetWindowTextA','JJCJ')
CALLDLL ('USER32','GetWindowTextA',#1,#S1,40) with
return value #2
OK message {[#S1]}

ComputerName

RegDll ('KERNEL32') with return value #5
CallDll('GetComputerNameA','C',#S1,'PJ',16) with return
value #5
OK message {Computer Names is [#S1]}

Get System Directory

Calculate #2 as 255
REGISTERDLL('KERNEL32','GetSystemDirectoryA','JCJ')
CALLDLL ('KERNEL32','GetSystemDirectoryA',#S1,#2)
with return value #2
OK message {System Directory is [#S1]}
Calculate #S1 as

Get Environment String

Calculate #S1 as
Selected procedures for
REGISTERDLL('KERNEL32','GetEnvironmentStringsA','C')
CALLDLL ('KERNEL32','GetEnvironmentStringsA') with
return value #S1
OK message {[#S1]}

Get Environment Variable

Calculate #S1 as
Calculate #1 as 255
REGISTERDLL('KERNEL32','GetEnvironmentVariableA','JCZJ')
CALLDLL
('KERNEL32','GetEnvironmentVariableA','PATH',#S1,#1)
with return value #2
OK message (High position,Large size) {[#S1]}

Get Windows Directory

Calculate #S1 as
Calculate #1 as 255
REGISTERDLL('KERNEL32','GetWindowsDirectoryA','JCJ')
CALLDLL ('KERNEL32','GetWindowsDirectoryA',#S1,#1)
with return value #2
OK message {Windows Directory [#S1]}
Calculate #S1 as

Reboot Windows

REGISTERDLL('USER32.DLL','ExitWindowsEx','JJJ')
CALLDLL ('USER32.DLL','ExitWindowsEx',2,O) with return
value #1

Get Volume Info

REGISTERDLL('KERNEL32','GetVolumeInformationA','JCCJJJJCJ')
CALLDLL
('KERNEL32','GetVolumeInformationA',"C:\",#S1,50,LONG2,LONG3,
LONG4,#S3,50) with return value LONG5
OK message {Volume Name for C:\\ is [#S1] it is [#S3]
formatted}

Find OMNIS Executable

Calculate SIZE as 150
REGISTERDLL('KERNEL32','GetSystemDirectoryA','JCJ')
with return value #F
CALLDLL
('KERNEL32','GetSystemDirectoryA',BUFFERNAME,SIZE)
with return value BUFFERNAME
Calculate DIRECTORY as BUFFERNAME
Calculate FILE as sys(1O)
REGISTERDLL('SHELL32.DLL','FindExecutableA','JCCC')
CALLDLL
('SHELL32.DLL','FindExecutableA',FILE,DIRECTORY,
RESULT_DIRECTORY) with return value
RETURNED_INTEGER
OK message (High position) {The OMNIS exe path and
name is : [RESULT_DIRECTORY]}

Log Drive Names

REGISTERDLL('KERNEL32','GetLogicalDriveStringsA','JJZ')
CALLDLL
('KERNEL32','GetLogicalDriveStringsA',255,#S1) with
return value #1
OK message (Large size) {[#S1]}

A more interesting example is a call to the Win32 functions to allow you to describe the size information of a hard disk drive.

A window was created as follows:

On opening the list is populated for the Drive selection as follows:

```
RegisterDll ('KERNEL32','GetLogicalDriveStringsA','JJZ')
CallDll('KERNEL32','GetLogicalDriveStringsA',255,DRIVE_STRING)
with return value #1
Set current list DRIVE_LIST
Define list {DRIVE_NAME}
While pos(chr(13),DRIVE_STRING)
   Calculate POS_13 as pos(chr(13),DRIVE_STRING)
   Calculate DRIVE_NAME as
mid(DRIVE_STRING,1,POS_13-1)
   Add line to list
   Calculate LEN_STRING as len(DRIVE_STRING)-POS_13
   Calculate DRIVE_STRING as
mid(DRIVE_STRING,(POS_13+1),LEN_STRING)
End While
Quit procedure
Local variable POS_13 (Long integer)
Local variable LEN_STRING (Long integer).
```

Upon selection the calls are made:

```
Parameter DRIVE_NAME (Character  10000000)
Calculate DRIVE_NAME as upp(DRIVE_NAME)
RegDll ('KERNEL32') with return value #5
CallDll('GetDiskFreeSpaceA','C',DRIVE_NAME,'
PJ',SECTORS_PER_CLUSTER,'PJ',
BYTES_PER_SECTOR,'PJ',NO_FREE_CLUSTERS,
'PJ',TOTAL_CLUSTER) with return value #5
Redraw numbered fields 1003 to 1014
RegisterDll
('KERNEL32','GetVolumeInformationA','JCCJJJJCJ')
CallDll
('KERNEL32','GetVolumeInformationA',DRIVE_NAME,
VOL_NAME,50,LONG2,LONG3,LONG4,
VOL_STOR_TYPE,50) with return value LONG5
Redraw numbered fields 1017 to 1018
```

The balance of the calculations on the window are are Display Field Calculations.

Stephen Miller
stephen_miller@dlagroup.com.au

Review of GUIDesigner v2.1

GUIDesigner is a GUI tool written by Mark Lowe of Stimulus Software. Mark is a excellent OMNIS programmer who is currently employed by OMNIS Software. Many of you who have seen the "Controlled Release" CDs can attest to the superior OMNIS knowledge of Mark as many of the examples were written by him. He has also been responsible for the examples that come with the Web Enabler Tools.

Mark has recently "free wared" the GUIDesigner product which is available for down load from: http://www.stimulate.com/software/gui/gui_designer.html.

You should make a point of downloading the manual as well as GUIDesigner is a complex product.

GUIDesigner works by establishing a relationship between Projects, an OMNIS library and a set of standards. The Standards can be one of the numerous standards that accompany the product or you can create your own. These standards are comprehensive and cover every GUI issue imaginable from background window colours to the alignment of data within fields. A series of setup windows allow you to adjust the standards to suit your business standards or to application standards.

One use of GUIDesigner which immediately comes to mind is to give an OMNIS 7v1.x application a facelift after conversion. I recommend that you take a converted 7v1.x library and create a project for it and apply the standard set of preferences. You can then attach all the windows in the application to the default standard or simply your main windows. You can attach or create a second standard to handle dialog boxes and the like. Choose 'New Project' from the File menu and then open the 'Windows Manager' as seen below.

You should then proceed to the Stylist window and apply the defaults you have set up globally

These simple tasks will give you an insight into the power of this product to save you a huge amount of time in the conversion of old libraries, into the enforcement of a corporate standard or the migration of third party product towards your GUI standards.

Features of GUIDesigner

Adjust and set background pattern settings per window object type.

Adjust and set colour settings per window object type.

Set the data alignment by field type.

A window object alignment tool for cleaning up the alignment of window objects.

A text formatting tool that enforces your standard, e.g, that the first letter of each word in button text be capitalised.

A tool that allows you to set up the 3d attributes of each window object, including the ability to apply GUIDesigner special 3d effects.

A tool that allows you to apply 3d effects to window rectangle objects.

The ability to create custom settings for fields and buttons to indicate that these fields are compulsory.

Buttons can be further referenced by name to add a particular button icon to each appearance of that button in your application.

The ability to modify the window attributes of windows globally.

The ability to create a standard colour table and export this table into any library.

The ability to apply standard window sizes and pixel margins to windows.

GUIDesigner is a valuable tool in the arsenal of any developer. Get a copy and spend a couple of hours checking out its features. GUIDesigner is supplied as a locked library and the source is available by negotiation with Stimulus Software.

Stephen Miller
stephen_miller@dlagroup.com.au

A Data File But No Library!

This could be a real life dilemma for you and raises some interesting problems. How do you extract the data from a data file when you do not have the library?

If all you wish to is report the data this is easy:
a)   Using the Shell logon as a SQL session to the datafile.
b)   Use the Interactive SQL Window to "Select * from Filename" to output all the data to screen.
c)   Print to file as tab delimited.

But what about if your client instructed you to rebuild the application?

One Approach would be to use the SQL FormBuilder and build a Window for each File Format (table). This approach relies on format variables to represent each field (column) and exclusively uses SQL coding.

Another approach is to use OMNIS SQL and notation to rebuild the file formats.

What are the OMNIS Equivalents of the Field (column) types returned by OMNIS SQL?

To Test this I created a file format with one field for each field type. I then used the following notation to interrogate the file structure:

```
Set current list #L3
Define list {NAME,TYPE,SUBTYPE}
Calculate #L3 as
$clib.$formats.OrigFile.$objs.$makelist($ref.$name,$ref.$objtype,
$ref.$objsubtype)
Redefine list {NAME,TYPE,SUBTYPE}
```

#L3 looked like this:

The Sub-types returned are correct. I was expecting character values according to the manuals but the numeric values returned do describe the Sub-types accurately. I then used the Make Window command to make a simple data entry window and entered a field records for my test file format.

I then closed the data file and opened it using SQL as follows:
```
Set current session {DLAT}
Start session {OMNISSQL}
```

Data File Tip

```
Set hostname {PM 6100:Desktop Folder:dla
test:testfile.df1}
Set batch size {10000}
Logon to host
If flag false
        OK message {Failed}
End If
```

I then tested creating the file formats in my library from the data file: Note I prefixed an "a" character to the file format name so my original file would not be overwritten.

```
Set current list #L1
Define list {#S1}
Describe database (Tables)
Build list  from select table
Calculate #L as 1
Repeat
        Load from list
        Calculate #S1 as con('a',#S1)
        Make file format from table {[#S1]}
        Save format {[#S1]}
        Calculate #L as #L+1
Until #L>#LN
```

Ok, so far so good... the next step is to try and map the Types and Sub-types returned by OMNISSQL to OMNIS types.

The Types returned by OMNISSQL (in #L2 below) are as follows:


```
Set current list #L1
Calculate #L as 1
Repeat
Load from list
Set current list #L2
Define list (Store long data)
{Name,Datatype,Column_Length,Decimal,Null,Index,Description}
Calculate #S1 as mid(#S1,1,90)
Describe table (Columns) {[#S1]}
Build list #L2 from select table
Calculate #S1 as con('a',#S1)
Calculate #L as 1
Repeat
Load from list
        If Datatype='SEQUENCE'     ;; Name
                Calculate #F as
        $clib.$files.[#S1].$objs.$add(Name,kSequence,,)
        Else If Datatype='DATE'     ;;
Name,Column_Length,Decimal
                If len(Column_Length)=0
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kDate,0,)
                Else
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name ,kCharacter,,15)
                End If
        Else If Datatype='DATETIME'
```

```
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kDate,1000,,)
        Else If Datatype='TIME'     ;;
Name,Column_Length,Decimal
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kDate,6,)
        Else If Datatype='VARCHAR'
                If Column_Length=0     ;; itemref
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kItemref,,)
                Else
                Calculate #F as $clib.$files.
[#S1].$objs.$add(Name,
kCharacter,,Column_Length)
                End If
        Else If Datatype='BOOLEAN'
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kBoolean,,)
        Else If Datatype='NATIONAL'
                Calculate #F as

$clib.$files.[#S1].$objs.$add(Name,kCharacter,1,Column_Lentth)
        Else If Datatype='INT'
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kInteger,0,)
        Else If Datatype='SMALLINT'
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kShortint,,)
        Else If Datatype='TINYINT'
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kInteger,32,1)
        Else If Datatype='NUMBER'
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kNumber,,1)
        Else If Datatype='FLOAT'
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kNumber,,1)
        Else If Datatype='PICTURE'
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kPicture,,)
        Else If Datatype='LIST'
                Calculate #F as
$clib.$files.[#S1].$objs.$add(Name,kList,,)
        End If
                Calculate #L as #L+1
        Until #L>#LN
        Save format {[#S1]}
Set current list #L1
Calculate #L as #L+1
Until #L>#LN

Local variable Name (Character   30)
Local variable Datatype (Character   50)
Local variable Column_Length (Character   50)
Local variable Decimal (Character   5)
Local variable Null (Character   10)
Local variable Index (Character   50)
Local variable Description (Character   150)
Local variable Cuurent_Session (Character   10)
```

The difficult bit here is choosing which is the correct DATE type and which is the correct NUMBERS type. The only way I can think to do it is to analyse the actual data and draw a conclusion on the basis of what the actual data corresponds to. Another possiblility would be to store all DATES as 15 chr length Character field and to treat NUMBERS similarly.

Well why don't you have a shot at it?

Stephen Miller
stephen_miller@dlagroup.com.au

Let's Get Enabled!

Since last November, we OMNIS developers have had the opportunity to enhance our applications via OMNIS' Web Enabler technologies, but many with whom I've talked feel too busy to travel the long and winding learning curve. So the question is "Which Web Enabler enhancements can help me satisfy my clients' needs, and how can I begin using them with a minimum amount of learning time?" Let's look at a couple of examples.

Put "Telepathic" On Your Resume

At Solid Solutions, all of our projects are client/server, mostly OMNIS/Oracle. We've found, as have many of you I'm sure, that end users don't always alert us to database errors they encounter, and we need these alerts "pushed" to us so that we can be proactively addressing problems even before we're contacted by the client.

If your users' environment includes an (increasingly popular) SMTP/POP mail server, then the Web Enabler extensions can come to the rescue. The *SMTPSend* command is a one-liner which, when placed in your common SQL error handling procedure (which you hopefully have), can send email to whomever you choose, in our case the OMNIS development team. You can include the database error information of course, but also as many of the client sys() values as you need provide a profile of the user workstation experiencing the problem.

Assuming that the Web Enabler extensions are installed on the client workstation, the command

```
SMTPSend
('MailServer','Sender','Recipient','TheSubject','TheBody','','','','','')
```

is all you need to add to your SQL error handling routine. *MailServer* needs to be an IP address or domain name, while Sender and Recipient each need to be a standard RFC 822 internet email address, e.g. rsweet@solidsol.com. Valuable things to include in the Body of the email message include sys() values for OMNIS version number, hardware platform, CPU type, available memory, and current session, as well as the procedure which was executing when the database error occurred. You can, of course, make this command a bit more robust by pinging the mail serverfirst with the TCPPing command, just to make sure it's available. The following example should do the trick:

```
Local variable lvResponseTime (Long integer)    ;; # of
milliseconds it took for the  pinged response to return to
me
Local variable lvPacketSize (Long integer) = 32    ;; how
big a packet to send
Local variable lvTimeoutThreshold (Long integer) = 500
;; # of milliseconds to wait before considering the ping a
failure
Local variable lvHost (Character   30) =
'204.167.66.111'    ;; the host I am attempting to ping,
either an IP address or a domain name
TCPPing (lvHost,lvPacketSize,lvTimeoutThreshold) with
return value lvResponseTime
If lvResponseTime<0    ;; negative number indicates a
TCP socket(WindSOCK) error.
OK message (High position,Large size,Sound bell) {Cannot
ping the server. Please call User Support.}
End If
Quit procedure
```

Be sure to supply a reasonable value for lvTimeoutThreshold (half a second in this example). Without it, OMNIS (Windows 3.x) hung when I tried to ping an unavailable host. Note that the WinSOCK error codes and their meanings are listed in both the Web Enabler manual, and in the on-line HTML documentation bundled with Web Enabler. Also note that you can use TCPPing to check the availability of the SQL DBMS server whenever a logon fails (assuming that you are connecting to the DBMS via TCP/IP). Why not let your application to do the detective work to help you quickly determine which communications layer is responsible for the failure?

Will email-enabling your SQL error handling routine give you that air of OMNIScience and mystery? It will at the very least allow you to post a sentry to help you provide proactive, quick problem turnaround for your end users.

Web/Database Fever

While the above email tip is a very easy one to implement, there's a lot more involved in using the web extensions to create an OMNIS-based common gateway interface (CGI) which acts as a web server, and allows web browsers to interact with SQL data. Focusing on OMNIS 7v3.5.x, let's review the *what* and *why* of this approach, and then we'll address the *how*.

Before I begin, however, let me emphasize that the terms "thick client" and "thin client" presented here have nothing in common with the OMNIS terms "thick client" and "thin client". As I use them here, they are standard terms in the world of internet/intranet web development, and their meanings are different. Therefore, toward the goal of understanding this article, please temporarily suspend your awareness of the OMNIS definitions.

The client/server model which most of us have followed involves a SQL DBMS, and a "thick client". By "thick client" I refer to a client Mac or PC which requires the OMNIS engine, one or more OMNIS libraries, and middleware such as Oracle's SQL*Net, to access the SQL DBMS. We all know how rich a user interface we can create with OMNIS. In cases where the host DBMS can't provide the server-side processing we need, the OMNIS procedural language also affords us the option of embedding business logic and other processing within the client application.

The major disadvantage of a thick client is the potentially high cost of client setup and distribution. OMNIS' CMS and Jim Pistrang's innovative "low tech" CMS both make distribution of OMNIS libraries easy. You can also script Fetch (Macintosh) or create simple FTP batch routines (Windows) to automate this as well. (Anyone wish to accept the challenge of writing a simple CMS using the Web Enabler FTP commands?)

Distributing new versions of OMNIS and the middleware, however, can require some manual steps, since Windows 95 and NT require Registry changes during most software installations. This can become costly if you have hundreds or thousands of users, or if end users are far away. Further, there may be dynamic data which you wish to make conveniently available to the world at large, and there is nothing as convenient as a web browser these days.

So how, you might ask, do we OMNIS developers provide access to SQL data from a web browser such as Netscape? The answer is to use Web Enabler's HTTP server commands to create a gateway. Instead of a two-tiered environment (SQL DBMS < > thick client), we now have a three-tiered architecture (SQL DBMS <> OMNIS CGI <> thin client). By "thin client", I refer to a client machine running only Netscape or Microsoft Internet Explorer to access the SQL data. Relatively little data processing (if any at all) is happening at the client, and there is no OMNIS engine, OMNIS library, or middleware to distribute or troubleshoot. Sounds like support heaven, doesn't it?

Three disadvantages of this thin client approach are:

Firstly, The user interface can only be as rich as HTML allows, i.e. a browser like Netscape. Secondly, A browser doesn't support any client-side data processing. And finally, Secure Sockets (SSL) are not supported by Web Enabler 2.0, but will be supported by a future release. SSL allows data to be encrypted for transport across the internet, and is essential for secure transactions.

Given these limitations, there are still many situations which beg for this approach. For those of you who agree, the question is now, "Given my 80 hour per week schedule, how can I get started as quickly and easily as possible?"

How About a Kick Start?

Web Enabler 2.0 includes an HTTP server tutorial library, which is a very basic OMNIS web server application, accepting requests from a web browser, and returning a web page. It is not, however, a complete application which ties an SQL database to the web. To help you get off the ground, I've created a "package" which demonstrates a complete, working link between an SQL database and the web. This package is modular, and includes a rewritten version of OMNIS' general purpose HTTP server tutorial library, and application-specific files which can be swapped out for different projects.

I decided to build this demo around Personal Oracle 7 database, although you client/server developers can modify it to work with any SQL DBMS. The demo logs onto Oracle as SCOTT/TIGER, and accesses the table DEMO.CUSTOMER, which comes bundled with Personal Oracle 7. The demo allows another computer to connect to this computer using a web browser, and search for customers via STATE or AREA CODE. It queries the database, downloads a results list, builds a web page on the fly (based on the results list), and sends that page back to the web browser.

My version of the HTTP server is a work in progress, and I have things to add, and many things to still test, before I put it in production. Please bear this in mind before you put it into production yourself, and of course feel free to contribute your own ideas and functionality.

This software is distributed as CharityWare. Any contributions sent to me for the use of this software will be forwarded to an anti-hunger charity such as Project Bread.

Now for an explanation of the pieces:

The entire package lives within a folder called CgiDemo. In here you'll find:

Shell.lbr – When launched, this library simply launches all OMNIS libraries in the folder called *Libs*. I saw Chris Clarke use this approach in his development of WebGate (Thanks, Chris!), and thought I'd try it here.

Libs - This folder contains the OMNIS libraries which together make the application-specific web server. These libraries are prefaced with a number to insure that they launch in the proper order:

1_WebSrv.LBR, which is the application-independent, generic HTTP server used for all . When this application starts up, it listens for requests from web browsers, and then brokers these requests to the appropriate application-specific libraries.

2_TigerWeb.LBR, which receives the web browser's request from *WebSrv*, processes it, and sends the results back to the browser . When this application starts up, it tells *WebSrv* which requests to send its way. When *TigerWeb* receives a request to search for customers for example, it

Converts the submitted web request into a SQL statement,
Queries the CUSTOMER table,
Downloads the results,
Converts the results to HTML,
Drops this HTML into a pre-constructed HTML template and
Sends the web page back to the browser

Html- This folder contains templates for all of the web pages which TigerWeb may need to send to a browser. Each template contains all of the "static" elements on the web page, along with placeholders for the dynamic, data-driven components. At the appropriate time, these templates are imported into an OMNIS field, and the placeholders are then replaced by the dynamic data by using the OMNIS *REPLACE* external command. Chris Clarke used a more powerful approach than this in his WebGate example, but my goal here was simplicity.

I've included comments generously, and in the Server Activity Log window have provided a bit more information than that provided in OMNIS' tutorial library. You can find the CgiDemo directory at ftp:/ /ftp.shore.net/members1/s/solid. Login as anonymous, and

download the entire CgiDemo directory – it's manageably small. You of course need to have TCP/IP set up on your network. network.ne

To run this demo:
1. Start Personal Oracle 7 on what we'll call your CGI machine.
2. Launch Shell.lbr on the CGI machine.
3. From another computer running a web browser such as Netscape, simply enter the CGI machine's IP address into the URL field, and hit Return. Note that on my Windows 95 laptop, I can run Personal Oracle 7, the CGI OMNIS application, and Netscape simultaneously. In Netscape, I simply enter the loopback address 127.0.0.1, which makes Netscape look for a web server on the same computer. Initial tests indicate that the Mac cannot talk to itself via a loopback address,but I await the good news from someone out there who has succeeded.)
4. You should now be presented with a Customer Search window, from which you can search via state and/or area code.

This demo's functionality is very basic, which should make learning time relatively short. Enjoy, and please let me know what you think and what bugs you find!

Ron Sweet
Solid Solutions, Inc.
33 Bedford Street, Suite 20
Lexington, MA 02173
Voice: +1 (617) 863-2775
Fax: +1 (617) 863-0540
Email: rsweet@solidsol.com
Web: http://www.solidsol.com/~solid

Ron Sweet is President of Solid Solutions, Inc., based in Lexington, Massachusetts. He has been an OMNIS developer since 1985, and is founder and co-director (with Jim Pistrang) of the Boston OMNIS User Group. He is also rarely responsible for the cuisine served at these user group meetings.

Ron Sweet

The OMNIS Advisor

Welcome to the third installment of the OMNIS Advisor. In this article I will discuss a simple architecture for controlling User access and functionality in an OMNIS application. At the end of the column, I will pose the task to be discussed in the next column. I encourage all readers to send me their solutions, comments, and questions at jim@jpcr.com.

The Task
How can I:
a) control access to OMNIS Windows, Menus and functions based on a User ID?
b) control access to specific functions within Windows?
c) build a simple interface to view and modify security settings? with a minimum expenditure of time and money?

The Solution
I have built a simple solution that works in the following manner:
1) Security access to a Window (or button, function, etc.) is determined by an access code number. In my example, a code of zero indicates no access is allowed, a code of 1 indicates view-only access, and a code of 2 indicates insert-delete-modify access.

2) Every Window (or button, function, etc.) that requires security is given a unique ID number, starting with 1.

3) Access codes for each user are held in a character string (char 100 in my sample). The string is stored in a USER table on the database, where the USER_ID is the unique key for the table. The table might look something like this:

user_id	access string
JIM	2210112021...
LBIRD	0112101000...

This indicates that JIM has modify access to windows 1 and 2, view access to window 3, no access to window 4, view access to window 5, etc. The user LBIRD has no access to window 1, view access to window 2, etc.
4) Access codes are viewed, inserted and modified via a User Maintenance Window. This window itself is controlled by the security system....typically a very small number of users are permitted to access this window.

What Really Happens
When a user initially logs on, their access string is fetched from the database and stored in the application...either in a MOFF or in a format variable that's maintained (not cleared) as long as the app is up & running. Whenever the user opens a window, a call is made to the Access Procedure, passing the unique ID number of the window, and returning the access code value for the user.

Based on the value returned, the window might not open at all, open in 'view mode', or open with all functions enabled.

The call to the access procedure looks like this:
```
Call procedure STARTUP/400
(fvAuthWin,fvWindowNumber) {check window security}
```
some comments:
fvAuthWin is a numeric format var. It will return from the call with the User's access code for the window.
fvWindowNumber is the ID number of the window being checked.

STARTUP/400 looks like this:
```
  Other parameters are optional
  Parameter pAuthWin (Field name)
  Parameter pWinNum (Short integer   (O to 255))
  Calculate pAuthWin as mid(gv.useraccess,pWinNum,1)
```
some comments:
gv.useraccess is a char 100 string that holds all access codes for the current user.
This call uses pWinNum to move the correct

View Mode and Modify Mode are handled in different ways, depending on the type of window involved. In the case of a modal window, the default attribute of the function buttons (Edit, Insert, Delete) is $active but not $enabled. When the window is opened by a user with a 'modify level' security access, a routine is called to set the $enabled attribute of the function buttons to kTrue. In the case of a modeless window, the default value of the editable fields is set to not $active. When the window is opened by a user with a 'modify level' security access, a routine is called to set the $active attribute of the fields to kTrue.

The User Maintenance interface displays the Access Code values for a selected user with a radio button interface. The string of 100 access codes is parsed into 100 format variables, which are displayed in a scrolling 'no row' OMNIS Table object. After the radio buttons are modified, the format variable values are parsed back in to the 100 character string for storage on the database.

Test the Demo
To best understand this discussion, download the JPCRDEMO.LBR from the following URLs:
windows: ftp://ftp.crocker.com/pub/users/pistrang/JPCRDEMO.zip
mac: ftp://ftp.crocker.com/pub/users/pistrang/JPCRDEMO.sea.bin

The file that you download contains a read-me, this article (as well as the prior Advisor articles), and two libraries, named JPCRDEMO and NEWVER. To test the access functions, just open the library!

Under the Hood
I've loaded the code with comments, so an OMNIS developer should be able to follow what's happening starting with STARTUP/0. Here are a few additional comments:
STARTUP/480 simulates a connection to a database and the retrieval of User information. In the demo the user ID is set to JIM and the Access String is hard-coded. Change the settings here to change your initial access settings.

The User Access Window (wmUser) simulates a call to a database in procedure 466. Change the values here to change the settings that you see when the window opens.
Use the 'Change My Settings' button on the User Access window to immediately change your current settings to those displayed on the window. Careful...if you change your settings to 'no access' and close the window, you won't be able to get back in!

Next Column
There have been some threads on the list lately concerning parameters and prompt windows. In my next column and demo I'll discuss using parameters to create some generic prompt windows, which can be called from anywhere within an application and return dates, numbers, and text. (I realize this was the 'next column' in my last issue, but it got bumped by the Security/Access column!)

Jim Pistrang.
jim@jpr.com
Jim Pistrang is the president of JP Computer Resources, based in Amherst, Massachusetts. He's a full time OMNIS developer, a Certified OMNIS Software Professional, an OMNIS Ambassador, and the co-director of the Boston OMNIS User Group. Contact him via email or visit him on the web at http://www.crocker.com/~pistrang/

George Schwalbe Joins DLA

George Schwalbe is the most recent addition to the DLA team. For the past three years he worked at Nortel Australia.

During this time his responsibilities have included:

Sole OMNIS developer for Australia/Asia for one year, assistance in two upgrades to Nortel's Human Resource database, and various improvements, customisations, and upgrades to over fifty windows.

Data Prime for Nortel World Trade during the 1996 Sales and Marketing/Management conference managing and designing a database to handle the hotel bookings and travel arrangements of over 700 delegates. The installation and testing of several digital telephone switching systems in the Sydney region, these include a GSM host and a remote switch for Optus, and the captive switch in Nortel Australia's training laboratory.

George has completed a Diploma of Computing studies from the University of Newcastle, and is currently pursuing a Bachelor of Business from UTS.

George manages front line OMNIS Technical Support and consults in an OMNIS development capacity to our Telecommunications clients. If you want to say hi, George's email address is george_schwalbe@dlagroup.com.au

Spam!

Spam can be one of three things; A disgusting experiment involving ham, and tin cans, an annoying Monty Python song, or the most feared of all three…electronic junk mail (he said shuddering with distaste…) Email is considered spam when it is sent unsolicited to many (often thousands) of recipients, or is posted to a large number of newsgroups (often where the particular posting is of no relevance.)

What is the true cost of spam? For the perpetrator, very little. Unlike traditional junk mail the sender will often pay nothing for sending out bulk emailings, for the recipients however, it is a whole different can of meat. When users pay for time on-line, they do not wish to see their time wasted downloading email from spammers. Similarly, host sites which receive and store the spam often pay for bandwidth by the megabyte and in turn past the cost of spam on to the end-users. All this time, whomever initiated the spam is sitting back waiting for the money to roll in having not spent much at all.

Spam in newsgroups increases the number of postings that subscribers read, as well as the space consumed on the new server machines at the service provider. As well, they are often posted to a special interest group for whom the particular posting is completely inappropriate to the newsgroup. For example, a post concerning cheap mobile phone rates sent to every programming newsgroup.

Similarly to the way paper junk-mailers buy mailing lists, spammers often automatically compile lists of email addresses simply by scanning newsgroups or web pages. Next time you post an article, you could become the target of a spammer! Perhaps there is no cause for alarm as there *are* steps which can be taken to send the message very clearly to the spammers that they are unwelcome. When people did not want paper junk-mail they would often return it to the sender (although I did read of a Canadian gentleman who subscribed to as many junk mail lists as he could for a constant free source of fire-starting material…) With spam however, the return address is often faked to prevent this, or more often, any reply to the email will return an automatic message to the effect of "thanks for your interest"…and thoroughly disregard your attempts to have yourself removed from the list. Occasionally, you can fool these mail filters by making your email sound genuinely interested in the product being promoted, for example "yes I am extremely interested"…in you taking me off your mailing list you leach! As great as the temptation to be abusive may be, most spammers will blissfully ignore such emails, if they even make it past the anti-abuse filters…

There are alternatives available if you are being covered with spam. The first course of action should be to send email to the administrator of the domain from where the mail originated. This is often postmaster@(wherever.) In so far as many spammers use bogus domains for initiating their spam, it may make sense to read the headers of the message and send CCs of your complaint to the administrators of domains further down the message's route that you know are legitimate. Often this will be enough to see the spammer given his marching orders as spam is often against the usage policies of service providers.

As so many spammers are selling snake-oil (no legitimate merchant or advertiser would stoop to this level) the spammer's arsenal often includes temporary email accounts such as the freebies given away by large service providers. As such, you will need to do some detective work to establish exactly which domains are real, and which are faked to avoid detection. This is not as hard as it sounds.

Say the email header contained the following lines:

     mail.bogusspamfiend.com
     server.technet.org
     mail.compucount.com
     mail.spammer.net

A traceroute of mail.bogusspamfiend would reveal no such host. Ditto server.technet.org. As soon as you find a legitimate domain, for example mail.compucount.com, start adding postmaster addresses to the CC of your complaint letter. If you are as mad as hell and you are not going to take this anymore, you could even write a script to automate the whole process.

If you have set up your email software to reject mail from certain domains, you can find a suggested "blacklist" of spammers domains at `http://www.cco.caltech.edu/~cbrown/BL` In the case of newsgroup spam, complaints should be sent to one of the Anti-Spam newsgroups such as news.admin.net-abuse.sightings.

If you are being unsuccessful with the above measures, there are a number of options available that are slightly more brutal than those suggested above. A number of Anti-Spam filters are also available from http://www-fofa.concordia.ca/spam.

If, however, you wish to make it your personal quest to rid the world of spam, there are a number of pre-emptive anti-spam techniques available. Web Poison, available from http://www.blackcat.net/cgi-bin/wpoison is a CGI that creates a web page containing thousands of made-up email addresses that the spammer's address crawler will add to its mailing list and dutifully follow, to no avail.

There are many groups who have been set up purely to deal with spammers. Try visiting http://spam.abuse.net/spam/ for an excellent list of ways to deal with spam effectively.

We can all do our own bit as well by ensuring we don't inadvertently become spammers by posting our messages to dozens of newsgroups (instead of cross-posting which allows the newsreading software to only display the message once.)

Meanwhile, I wonder if I can burn my mail server to keep the house warm…

Spam's not Kosher

Daniel Lewkovitz
daniel_lewkovitz@dlagroup.com.au

OMNIS 7v3.5.4 and 3.6

Release of OMNIS 7v3.5.4

Because this is not a general maintenance release, and because the bugs that are fixed therein are not something that a majority of our customer base will need, the decision has been made to send this release out to only those customers who need it or who specifically request it. You all are able to request a copy of this release (if you have Software Support), by sending an email message to OMNIS_support@dlagroup.com.au

This message has been posted on the DLA Web page and the US List Server, with the data of what fixes the release contains, so that developers will know whether they need this maintenance release or not.

Release of OMNIS 7v3.6

This new release has just entered beta 1 testing and the full features are not settled. Australasian Developers with DLA Technical Support may apply to join the beta programme by contacting George Schwalbe at DLA on george_schwalbe@dlagroup.com.au.
If there are any specific features/fixes which you desire in v3.6, please forward them to George as well and they will be considered for this and future releases.

What's been fixed in 7v3.5.4:

Fault number/description

PR/@B/007 — Running a select statement twice to SQL Anywhere from the W95 version of OMNIS will cause OMNIS to crashif it contains a date field.

PR/@B/010FC — Memory leak associated with "reset cursor(s)" command and the ODBC DAM

PR/@B/008FC — OMNIS for Win95 crashes selecting a date field connected to DB2 via ODBC

PR/@B/027FC — SQL Server 6.5 support via the ODBC DAM

PR/@B/015FC — Request for Power Mac native O7 ODBC DAM

PR/@M/011FC — GPF when doing a GROUP BY on long integer columns

PR/@O/027FC — OMNIS customers request the ability to use SQL*NET 2.3x on MS-Win OS

PR/@0/004FC — Customer requests that Oracle DAM have no dependencies on files found only in SQL*NET v1.1

PR/@O/050FC — Support for SQL*NET 2.3 on PowerMac (it's still 2.0 for Mac)

PR/@O/041FC — Win95 OMNIS crashes when doing selects from Oracle with a 13-character user account name

PR/@O/015FC — Decode function consistently returns null when executed in the select clause of a query

PR/@O/028FC — When logging on to Oracle 7.3 with a 30-character user name, GPF occurs when the SQL script is executed.

PR/@S/008FC — Customers request that a PPC-native version of the Sybase DAM be made available

PR/@S/011FC — Customers request that the Sybase DAM be made compatible and certified with PPC-native Open Client v.10.0.4

PR/@S/013FC — Support for Sybase System 11

PR/CP/016FC — Mac OS 7.6 compatibility

PR/NT/126FC — Customer is requesting a library preference to allow Nulls to come back as Empty from a back end database

PR/FU/007 — Decode function causing insufficient memory - linked to PR/@O/015FC which is fixed

PR/CP/015FC — NT 4.0 support

PR/ET/190FC — Update the Lotus Notes extension for compatibility with v4 of Lotus Notes

PR/@O/012FC — Fetching from select tables formed by large, complex multi-union select statements results in system crashes

PR/@O/053FC — Long select statements with many calculations cause "out of memory"

PR/@O/013FC — After a certain number of insertions and deletions to "wide" tables (c. 60 cols) client Power Macs freeze

PR/AH/012FC — Calculated fields will not subtotal in the ad hoc report writer

Informix DAM

OMNIS

the omnis underground communique

www.OMNIS-underground.com

The OMNIS Underground and it's membership proudly sponsors the OMNIS list server. This free daily communications forum for the OMNIS developer is open to all who wish to subscribe. Currently boosting a subscribership of over 500 developers from around the world, this service provides some of the best technical tips, advice and open discussion of any community on the Internet. To join this service simply send a message to: LISTSERV@OMNIS-underground.com

The body of the message should read:
subscribe OU_DROP_POINT

You will then receive a mail welcoming you to the list and providing additional instructions on how to use the list.
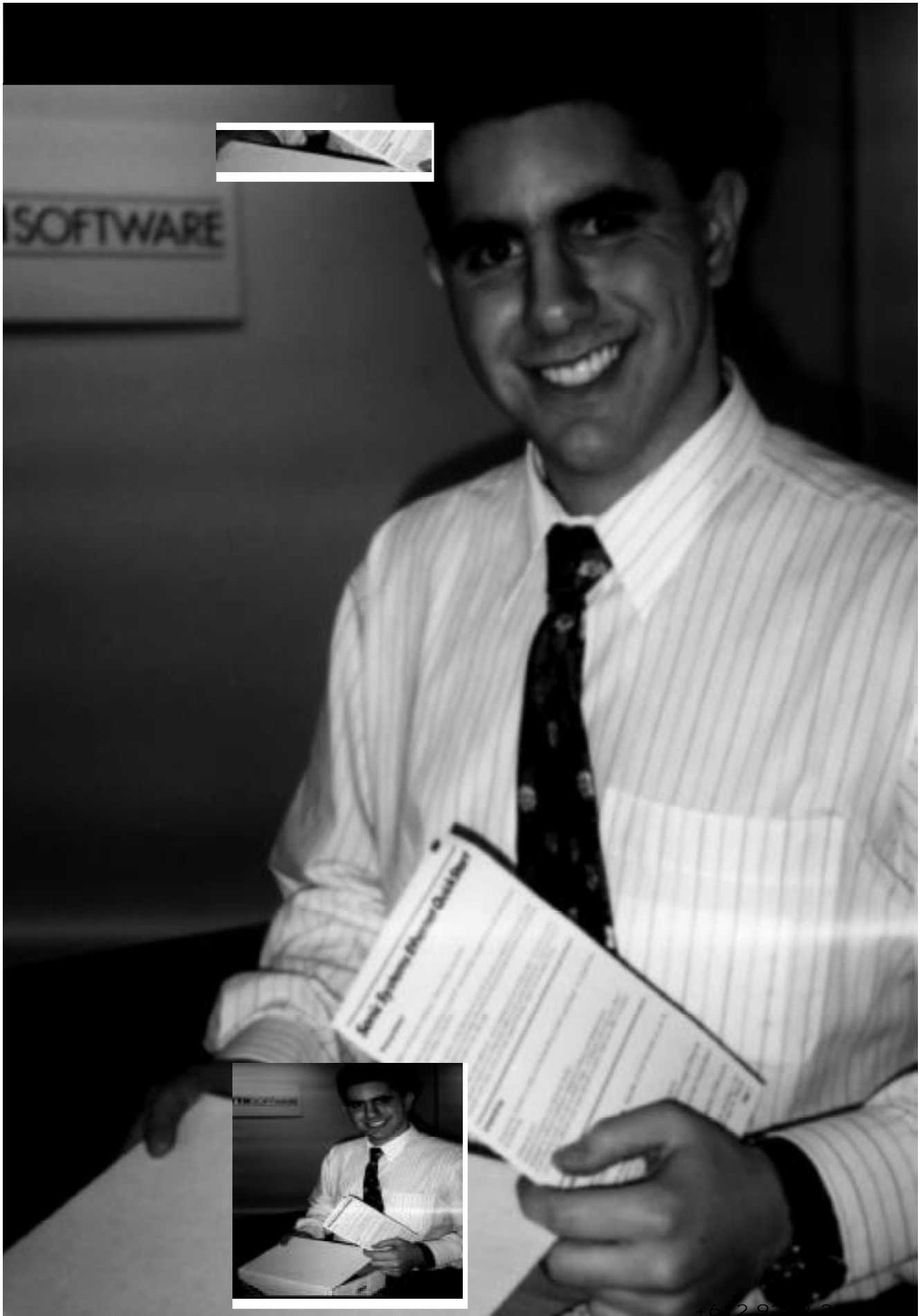
We openly invite you to join and share your knowledge and questions with the community.

The OMNIS Underground's Web site is designed to be a resource for you and the community. The site includes information about the Underground and it's activities. It has full FTP services and it's archives include many of the lecture notes from the conferences, sample libraries, a code repository and much more. This site is designed to help you and the OMNIS development community share information and tools. If you would like to contribute to the site or have suggestions please let us know.
The URL is http://www.omnis-underground.com.

Underground Technical Conference Video Tapes

Video Tape Packages by Track	Tape Numbers	Package Price	Underground Member Price
Prometheus Track - 12 tapes	PR-1 to PR-12	$319.00	$239.25
Internet and Web Dev Track - 10 tapes	AI-1 to AI-10	$269.00	$201.75
Omnis 101 Track -12 tapes	OM-1 to OM-12	$319.00	$239.25
Advanced Omnis - 14 Tapes	AO-1 to AO-14	$369.00	$276.75
Total Price for All 48 Tapes: The above 4 tracks = **What's your time worth? Buy all tapes and save a bundle!!!**		$1276.00	$957.00
Buy all 4 tracks & get Package 7 for the bonus price of:		$1195.00	$995.00
Over 90 hours of instruction! 56 tapes in total!			
Special Video Tape Packages			
1. SQL Track - 8 tapes	AO-1 to AO-8	$239.00	$179.25
2. Notation & Mysteries-6 tapes	AO-10 to AO-14 & OM-9	$179.00	$134.25
3. Beginning thru Advanced SQL - 3 Tapes	AO-1 to AO-3	$99.00	$74.25
4. Oracle Exposing the Giant- 3 Tapes	AO-5 to AO-7	$99.00	$74.25
5. Myths and Mysteries - 2 Tapes	AO-13 to AO-14	$69.00	$51.75
6. All Notation Tapes - 4 Tapes	AO-10 to AO-12 & OM-9	$129.00	$96.75
7. Prometheus Track from 1st Conference - 8 Tapes	PROM-1 to PROM-8	$169.00	$126.75
Buy any 2 of the Special Video Tape Packages (#1-7) and get the third for 1/2* price!!!		All prices are in $US	