TODAY'S NIAKWA NEWS

Vol. IV, Issue 1 COMMUNITY NEWS AND INFORMATION January 1995

INTRODUCING...
GUI FEATURE OF THE MONTH SDK

For some time we have heard how many NPL applications beat their competition in features and functionality, but lose on "look and feel". At our October 1994 Worldwide Conference, we heard that it was time Niakwa did something about it.... and we have. We are pleased to announce the Niakwa Screen Manager Software Developers Kit (NSM-SDK) that will provide NPL developers the following benefits:

- NPL applications will have the option of adding a true MS-Windows interface (MS-Windows has won the user interface battle for now.)
- A text based user interface will also be supported for non-Windows environments from the same set of source code!
- Adding the Windows look and feel will be easy through the use of a visual editor and Niakwa's interface to the MS-Windows API calls.

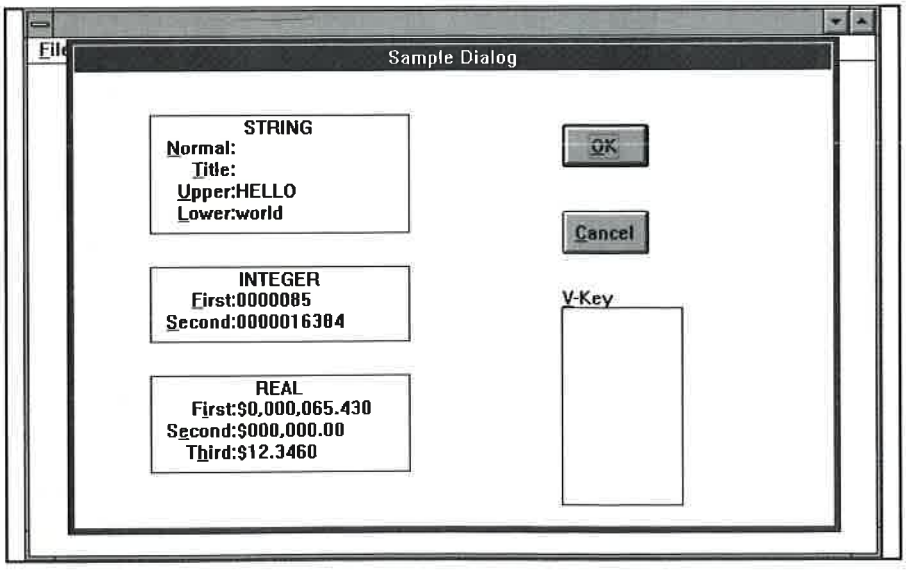
In an effort to get Windows capabilities in your hands as quickly as possible, we have crafted the GUI Feature of the Month SDK Program whereby we will roll out a basic version of the screen manager this month and then add new features on a monthly basis.

The first release of the Niakwa Screen Manager will use a text based resource file and resource compiler.

Very shortly, we will be providing a true visual editor which will make screen management even easier to work with and modify.

The initial release will contain all of the features listed in the box to the right.

Niakwa is committed to delivering a "developer friendly" screen management solution that will allow you to compete against MS-Windows applications' look and feel while also maintaining the true portability. Because this is an ongoing commitment, we welcome your input on additional features you require for the future. We encourage you to sign up for the "GUI Feature of the Month Club" and be well armed for the MS-Windows battlefield for sales. The Windows screen which you see on our cover is an example of what you can do right now!



NPL Programs get Common User Interface compliance with NSM!

GUI Feature of the Month January 1995

Dialog Box

A window that appears temporarily to supply or request information

Drop Down Menu

A list of available commands in an application window

Button

A small box used to request an action to take place

Scrollable List Boxes

A type of box that lists available choices, including options beyond what is visible in the box

Static Text

Text which is permanent to the screen display

Edit Text

Text which can be input or overridden

Mouse Support

Mouse supported as currently available on NPL platforms

ASK ANDY

Q: I have converted most of my users to NPL Release IV. I have several users on IBM RS/6000 systems and would like to move them to Release IV also. Is Release IV available for the IBM RS/6000?

A: NPL Release IV is currently available for all Niakwa supported MS-DOS based platforms (MS Windows, Pharlap, Novell, NetBIOS) and Intel UNIX platforms. All RISC versions of NPL are in the process of being ported to Release IV. A firm Release date for these products has not been set, but you can expect to hear more about the availability of these products in the near future.

Q: Can NPL modules coexist with other applications?

A: You bet! It's easy to code module routines for existing applications. Once you have a working module, you must first include it in your current application, (i.e.; 0010 INCLUDE T My_Module). When your program is loaded, your new module is loaded into a separate workspace and is referenced via NPL's FUNCTION call interface (far superior to GOSUB/DEFFN). For a broader overview of what's new in NPL Release IV, check out Chapter 4 of the NPL Release IV Programmer's Guide, or contact your Niakwa Sales Representative.

RELEASE IV UPDATE AVAILABLE

As of January 1, 1995, Niakwa is shipping an updated version of NPL Release IV for all new and upgrade orders. This is a copyable update file which can be used to update all prior Release IV versions. NPL developers may download updated executables from the Niakwa BBS and update their existing Release IV sites with these new executables.

Several new features have been introduced into the language. The following summarizes several of the new features included.

- Provisions for enhanced file access for Novell and NetBIOS environments.
- Mixed NetBIOS/Netware network support
- Support for more than three printers under Novell
- Internal enhancements required to support the new Windows 2227 Communications Driver.
- A new option ("-ERRFORMAT") . This option controls the format of error diagnostic lines.
- Support for early discard of restricted memory and large uninitialized variables in memory.

In addition to these enhancements, NPL Release IV revision 4.10.23.xx.x also corrects several bugs discovered through internal testing at Niakwa as well as those reported from the field. Bug Report #11 is now online on the Niakwa BBS for review.

A complete description of all new features implemented in this revision of NPL is included in all new NPL Release IV Supplements as NPL Revision 4.10.23.xx.x Update Release Notes. In addition, these Release Notes and executables are maintained on the Niakwa BBS for your convenience, or can be ordered on diskette.

NIAKWA ANNOUNCEMENTS

Niakwa is, and always has been, an exciting place to work. Two key factors make this so: one is the high level of energy and commitment of our staff, which we refer to A-C-H (Attitude, Commitment, and Humor). Another is the fast changing nature of the computer industry. Responding to those factors has provided new opportunities within our group.

Dick Drew and Andy Warzecha are involved in developing a second and third revenue "leg" to the Niakwa business: Imaging and Custom Programming Services. By supplying these services, we are able to strengthen the NPL development environment by gaining expertise in the fast growing imaging market, and by making additional programming talent available to NPL developers for assistance in enhancing and modernizing their application code.

Effective immediately, **Cyndee Philyaw**, International Account Manager and ten year veteran of Niakwa, has been promoted to NPL Sales and Marketing Manager.

Kurt Skaronea, seven year worldwide support guru, has been promoted to Manager of NPL Technical Services.

Jan Strickland, (five years), that "behind the scenes" force that makes all things happen... particulary where NPL software meets diskette, has been promoted to Production/Shipping Supervisor.

Debbie Benson, who has patiently taken your orders for two years, is assuming new Sales and Marketing responsibilities. And last but not least, **Joyce Craig**, that friendly voice you all know and love, is replacing Debbie at the NPL Order desk.

Our newest friendly voices are **Patricia "Tish" Keating**, who joins us as our new Receptionist and Administrative Assistant, and **Valerie "Val" Lau**, our new Production and Shipping Assistant.

And, at our research and development office in Winnipeg, Canada, we welcome **Dale Cantafio** and **Rob Dyck**, who will be responsible for completing Release IV for the IBM RS/6000, HP 9000, and SUN RISC platforms. Dale is a specialist in C programming and data communications. Rob has broad expertise which includes dBase, Powerhouse, and C applications, as well as databases and network software.

We are confident that these changes will be good for you as well as us.

Window Dressing....with the NPL Gateway to MS-Windows API Library

Editors Note: As usual, Niakwa has provided you with more than one option to meet your need for a true Windows Look and Feel. Our GUI Feature of the Month article relates to a new screen management product (NSM) which provides a solution for both Windows Graphical programs and non-Windows text based programs. The Niakwa Screen Manager provides a Windows user interface which will be a good solution for most developers, and also a developer friendly product. However, for those developers who need to get deep into the entire Windows API functionality, we offer the NPL Gateway to Windows API solution. This article pertains to this second alternative; a solution targeted solely to Windows platforms. The MS-Windows API product is not as easy to use as the Niakwa Screen Manager. But for those who require the flexability of the full Windows API, there are ways to simplify the start-up time involved and get you off on the right development track. The following are several examples designed by our Winnipeg staff as their recommended way of adding a few Windows looks to your applications in short order. More complete examples are included with our NPL Gateway to MS-Windows API product (available now as a library class product!). Please contact your authorized Niakwa Reseller or Niakwa directly for details.

What exactly is the NPL Gateway to MS-Windows?

Every NPL Programmer today has the ability practically at your fingertips to add Windows capabilities to your NPL application!

In April, 1994, Niakwa introduced the NPL Gateway to MS-Windows API library package to allow writing full-blown event-driven windows applications in NPL. But, you may say, learning the entire Windows API can be a time-consuming and confusing process. Isn't there an easier way to make my application look like it belongs in the Windows environment?

Well, there is! Much of the complication of the Windows API comes from the Windows standard "document/ view/ menu" display convention. But you don't have to slavishly follow this convention. In fact, many applications written using Visual Basic use a more natural "forms-driven" approach.

How should I begin?

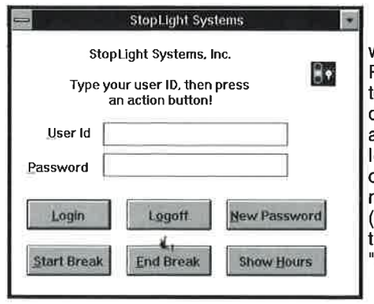
Starting with a completely blank application window and-menu of options can confuse a naive user - how should THEY know to select Open/File and the location and name of the document? Why doesn't the program just ASK?

In such a situation it may be more natural (especially for NPL programs) to start by displaying a dialog box, with buttons or other information that will lead to other dialogs, and eventually to one of the useful jobs that the application was designed to do.

What does it take to use this "dialog-driven" approach with your NPL application? With the NPL Gateway to MS-Windows API, it can be done very simply:

- Design the dialog box layouts using a tool that allows you to save them in a .DLL file. Borland's Resource Workshop is excellent for this. Be sure to give the buttons and other controls that you need to reference in NPL names. This will produce an "include" file designed for C programs.
- Run the "npldefs" program to convert the include file into an NPL .SRC file, and compile this into your application diskimage.
- You can now make the dialog "pop-up" using the "DialogBox" function call.
- You will need to write a short dialog box procedure to handle events, like button presses. Eventually, one of these events will cause the dialog box to go away, and you call the "EndDialog" function call.

Can you show me an example?



A simple example will help. I use Turbo Resource Workshop to design the sample dialog , with an icon, a title, buttons to select one of several options, and a login name and password (see figure) and put this in a file called "simplex.dll".

This also generate a C format include file "simplex.h" that looks like:

```
#define BUTTON_PASSWORD    102
#define BUTTON_LOGIN       106
#define BUTTON_BREAK       103
#define BUTTON_END_BREAK   104
#define BUTTON_HOURS       107
#define BUTTON_LOCOFF      105
#define EDIT_USERID        101
#define EDIT_PASSWORD      110
#define STOPLIGHT          100
```

I use the command "npldefs simplex.h simplexh.src _" to make an NPL version of this file that looks like:

```
1000 PUBLIC:;symbols from simplex.h
   : DIM _BUTTON_PASSWORD=102
   : DIM _BUTTON_LOGIN=106
   : DIM _BUTTON_BREAK=103
   : DIM _BUTTON_END_BREAK=104
   : DIM _BUTTON_HOURS=107
   : DIM _BUTTON_LOGOFF=105
   : DIM _EDIT_USERID=101
   : DIM _EDIT_PASSWORD=110
   : DIM _STOPLIGHT=100
   : END PUBLIC
```

NIAKWA™

The code to display the dialog box looks like:

```
0000 INCLUDE T"WINDEV"
   : INCLUDE T#_WINDEV,"WINDOWS"
   : INCLUDE T#_WINDEV,"OPTIONS"
   : INCLUDE T#_WINDEV,"CSTRING"
   : INCLUDE T"SIMPLEXH"
   : ;
   : FUNCTION
      'SimplexDialogFunc(user-
      instance,hDlg,wMsg,wParam,
      lParam)/PUBLIC/FORWARD
   : PROCEDURE 'Cleanup/EXIT
      /FORWARD
   : ;
   : DIM Ignore,button
   : DIM hInst,lpDialogFunc
   : ;
   : DIM Simplex$32,Startup$32
   : DIM UserId$32,Password$32
   : ;
0100 'DisablePoll
   :Simplex$="SIMPLEX.DLL"&HEX(00)
   : Startup$="STARTUP"&HEX(00)
   : ;
   : hInst='LoadLibrary(Simplex$)
   : lpDialogFunc='MakeNPLProc
      Instance($NAMEOF(FUNCTION
      SimplexDialogFunc),
      _CALLBACK_DLGPROC,0)
   : ;
   : UserId$=" " :;start blank
   : PassWord$=" "
   : ;
   : button='DialogBox(hInst,
      Startup$,_NULL,lpDialogFunc)
   : ;
   : ;process result based on
      button here
   : ;free resources
   : 'Cleanup
   : STOP
   : PROCEDURE 'Cleanup/EXIT
   : IF lpDialogFunc<>0
   :    'FreeNPLProcInstance
         (lpDialogFunc)
   :    lpDialogFunc=0
   : END IF
   : IF hInst=_HINSTANCE_ERROR
   :    'FreeLibrary(hInst)
   :    hInst=0
   : END IF
   : 'EnablePoll
   : END PROCEDURE
0200 FUNCTION 'SimplexDialogFunc
      /BEGINS
   : DIM x
   : SWITCH wMsg
   : CASE _WM_INITDIALOG
   :    ;set dialog controls that
         are modifiable
   :    GOSUB set_dialog_info
   :    'SetDlgItemText
         (hDlg,_EDIT_USERID,
         'Cstring$(UserId$))
   :    'SetDlgItemText
         (hDlg,_EDIT_PASSWORD,
         'Cstring$(Password$))
   :    RETURN(_FALSE):;focus
         not set
   : CASE _WM_COMMAND
   :    GOSUB get_dialog_info
   :    SWITCH wParam
   :    ;all buttons exit the
         dialog
   :    CASE _BUTTON_PASSWORD,
            _BUTTON_LOGIN,
            _BUTTON_BREAK,
            _BUTTON_END_BREAK,
            _BUTTON_HOURS,
            _BUTTON_LOGOFF
0300       'EndDialog(hDlg,wParam)
   :    CASE
0400       ;edit controls also get
            messages,when focus
   :       ;obtained/lost, etc
   :        RETURN(_FALSE)
   :       ;message not handled
   :    END SWITCH
0500 CASE
   :    RETURN(_FALSE):;message
         not handled
   : END SWITCH
   : RETURN(_TRUE):;message
      handled
   : =set_dialog_info
   :    ;set dialog controls that
         are modifiable
   :    'SetDlgItemText(hDlg,
         _EDIT_USERID,
         'Cstring$(UserId$))
   :    'SetDlgItemText(hDlg,
         _EDIT_PASSWORD,
         'Cstring$(Password$))
   :    RETURN
   : =get_dialog_info
   :    ;get dialog controls after
         modification
   :    x='GetDlgItemText(hDlg,
         _EDIT_USERID,UserId$,
         LEN(STR(UserId$)))
   :    STR(UserId$,x+1)=" "
   :    x='GetDlgItemText(hDlg,
         _EDIT_PASSWORD,Password$,
         LEN(STR(Password$)))
   :    STR(Password$,x+1)=" "
   :    RETURN
   : END FUNCTION
```

To Recap:

That's really all there is to it! When you call 'DialogBox, a popup dialog appears the way you designed it, and then goes away when a button is pressed.

After calling the 'DialogBox function, your application can use the values provided by the user (in UserId$ and Password$ and the button number) to proceed the way it normally would, displaying results in the NPL window.

Alternative Styles:

Of course, there's a lot of other ways the 'DialogBox function can be used. You don't have to wait for the Dialog box to finish to use the values in the dialog, and you don't have to end the dialog just because a button was pressed.

Or, some buttons on the dialog box might also be used to display other dialog boxes, if more information is required for a specific option. It's usually a good idea not to 'nest' dialogs too deep, though.

You can do almost anything you want while in the dialog box callback function, but keep in mind:

- You MUST return a value from the dialog procedure that indicates (except for _WM_INITDIALOG) whether you have processed the message.
- You may not 'bail out' from a dialog procedure via program overlays, and you should not attempt to modify code while the dialog box is displayed. The dialog box NEEDS your code. Close the dialog box first, THEN make program changes.

Output Displays as Dialogs:

You may decide to convert your output displays to dialogs as well. Many of these could be replaced by dialogs with a bunch of 'static' controls or list boxes showing the results and a few buttons indicating the available options (continue? cancel?). When you are converting existing applications that PRINT the information to the display, INPUT SCREEN can be a useful way of getting the printed results from specific locations on the screen (this is a technique known far and wide as "screen-scraping").

The more user data you obtain and display via dialog boxes, the more your application will look like a 'native' Windows application.

Summary

As you can see, the NPL Gateway to MS-Windows API has a lot to offer the developer who is serious about needing a true Windows look and feel, including full Windows API access, and where source code compatability is not the primary concern. Detailed examples come with this library and training classes can be arranged.

Note: For the developer who requires the modern Windows look, ease of use, and source code compatability, we recommend the new Niakwa Screen Manager (NSM) Product announced in January, with an SDK version available now.

Editors Note: The coding example detailed in this article and two additional examples described above are available for No Charge on the Niakwa BBS.

Niakwa, Inc., 23600 N. Milwaukee Ave., Mundelein, IL 60060 Phone (708) 634-8700 FAX (708) 634-8718

Getting Started With Release IV

by Craig Freeman

I clearly remember the day that Niakwa's Release IV 'beta' kit arrived. Every other Niakwa revision had been a 'slam dunk' to implement so I set aside a single evening to look things over and to write some nifty NPL code. After reviewing the manual for about an hour, my first thought was..."I am never going to understand this stuff!"

Words like"LINs" and concepts like "modules" and "variables scoping" read like Greek. Furthermore, there were memory issues and potential upgrade costs to consider. Just the idea of spelling out "BalanceDue" instead of "B7" meant going back to school for a refresher typing course. My existing utility programs would soon crash on long identifiers. I found every reason in the world to toss that SDK in the trash until a simple fact hit me...for the first time in nearly two decades of writing BASIC-2 code, it wasn't the computer that had become obsolete--it was me! When I finally realized that NPL Release IV is more like "C" than BASIC, I dusted off my copy of Borland's C++ Introductory Guide and read for a while. Once I had a basic understanding of libraries and modules, my initial fears subsided. Soon after, the benefits of Release IV became obvious. I have seen the light! And, so can you.

Making the switch to Release IV...

Follow the guidelines below and you'll be writing in Release IV in no time. All you'll have to do is keep the memory requirements down and get all your users to pay for an NPL upgrade. Believe it or not, that's easier than it sounds. The memory issue turned out to be a non-issue. Converting my most commonly used subroutines to procedures saved us many thousands of bytes in memory. With our largest programs, we used the enhanced MAT REDIM statement to adjust the program size to available memory.

Windows, PharLap, and UNIX had near unlimited memory. Most of our 386 PC users got by with adding the /M and /U options to their RTI calls. The really stubborn cases (mostly "286" PCs) were resolved with DOS upgrades. And, since NPL Release IV code tends to be even more stable than previous versions, we weren't nearly as troubled with using RTP (for an 80K savings) as we were in the past. The dreaded memory issue was not so bad after all.

Upgrading your End-Users...

We wanted to maintain one set of code in Release IV and to abandon all previous versions forever--something that would be impossible unless we could get almost all of our clients to pay a small but not insignificant upgrade fee. Our strategy was simple and it has worked to everyone's benefit:

"Make it a rule to do all new software development in Release IV."

- Make it a rule to do all new software development in Release IV. Any time we add a new function to one of our core products, we advertise it to our clients with a notation that it requires NPL Release IV. To avoid problems, our menu software always tests $REV and refuses to load NPL functions if they haven't upgraded.

"Include the upgrade cost in every customization quote."

- Include the upgrade cost in every customization quote. It takes less time to write or modify code using Release IV and it is less likely to produce bugs. On any fixed cost custom quote, we include the Run-Time upgrade as a necessity. Overall, we (and our clients) are better off.

"License or create something wonderful that is worth the upgrade fee."

- License or create something wonderful that is worth the upgrade fee. We analyzed our client's needs and found a nifty idea ("Vista") that would be easy to write in Release IV. We then informed our clients that we would give it to them for free if they paid for the upgrade. We make as much on the NPL upgrade as it costs (not to mention the added support revenues). Our clients are thrilled. Nice. Very nice!

With all of the above, the majority of our core product users have already upgraded (most of the remainder being inactive clients for whom all of our development and maintenance efforts would be wasted in any case). As a result, we are now free to enhance our products and stay ahead of our competitors.

So much for that feeling of becoming obsolete! And...so much for a Sunday night when I was overcome with altruism as I prepared a list of helpful hints for converting your Release III code to Release IV. This document is available to you at no charge on the Niakwa BBS in the Developer's forum as file NPLHINTS.ZIP. The decision to move to Release IV (if you haven't already) is definitely worthy of a high place on your 1995 Resolutions list!

Editors Note: This article originally contained two additional pages of Release IV coding hints to help any developer preparing to code in Release IV. These helpful hints are available for No Charge on the Niakwa BBS.

Niakwa BBS Listing

For your convenience, the following new information is available on the Niakwa BBS:

- **Current Product Revisions Listing**
- **Rev 4.10.23.xx Maintenance Release**
- **Rel IV Helpful Hints: NPLHINTS.ZIP**
- **MS-Windows API Ex: WAPIEXAM.ZIP**

To access the BBS, dial 1-708-634-6227. Set your communications parameters to 8 bit, no parity and 1 stop bit. The BBS adjusts automatically to baud rates up to 9600 baud. First time users - login as GUEST and leave a message to the SYSOP requesting a personal login and password. Your personal login and password will be provided to you during normal business hours.

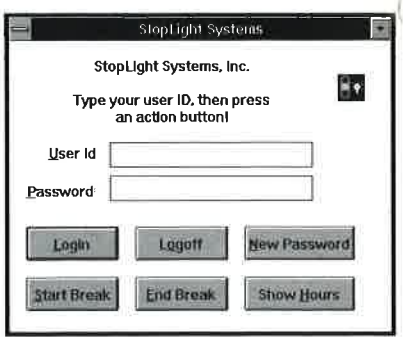
Niakwa, Inc.
23600 N. Milwaukee Ave.
Mundelein, IL 60060

COMMUNITY NEWS AND INFORMATION

Inside This Issue:

And Special Insert:
Window Dressing with NPL Gateway to MS-Windows API

See *Special Insert* for instructions on using the NPL Gateway to MS-Windows API to generate dialog boxes like this one!



Today's Niakwa News

EDITOR: Cyndee Philyaw
STAFF WRITERS: Joe Brekelmans, Pat Legg, Jay Routson, Andy Warzecha
CONTRIBUTORS: Craig Freeman

Today's Niakwa News is published periodically by Niakwa, Inc., 23600 N. Milwaukee Avenue, Mundelein, IL 60060. Phone (708) 634-8700. Fax (708) 634-8718.

Comments, questions, and suggestions can be directed to the Editor, Today's Niakwa News, Niakwa.

Worldwide Conference Feedback

Here's a sampling from the numerous letters received following our conference:

"It had just the right mix of information and fun. We learned a lot and came away with enthusiasm for all the new possibilities open to us."
Kathy and Jim Palladino
Basic Software, Inc., NY

"It is obvious you believe in the A-C-H (Attitude, Commitment, and Humor) approach to life. You have a great staff, a great set of products, and a wonderful vision for growth."
Don Youngberg
Hill Arts and Entertainment Systems, CT

"Thanks for a GREAT conference. I may have to write three ot four thousand lines of NPL just to be settled down enough to sleep. You enthusiasm for the future of NPL and the energy of your staff are reassuring and motivating."
Sam Matzen
Matzen Computer Resources, KS

"The talents, support and teamwork of your staff have been an inspiration since we purchased our first RunTime program ten years ago."
Dan Niesen
Anasys, Inc., WI