NIAKWA VISUAL INTERFACE MANAGER RELEASE V ADDENDUM

1st Edition - August 1998

Niakwa, Inc.
23600 N. Milwaukee Avenue
Vernon Hills, IL 60061

TEL: (847) 634-8700 FAX: (847) 634-8718 URL: http://www.niakwa.com

Table of Contents

This page intentionally left blank

CHAPTER 1

VINNY ENHANCEMENTS

1.1 Overview

This addendum discusses all updates and enhancements introduced in Release V of the Niakwa Visual Interface Manager (Vinny).

Section 1.2 discusses the contents of Vinny Revision 5.0.

Section 1.3 discusses installation notes related to the MS-Windows Setup procedure.

Section 1.4 discusses upgrading projects to Vinny Revision 5.0.

Section 1.5 discusses updates and changes to Instant Vinny.

1.2 Contents

The Niakwa Visual Interface Manager (formerly Visual NPL) is the graphical user interface builder of the Niakwa IDE. The primary change to Revision 5.0 is Vinny has been updated with 32-bit support and is now bundled with the Instant Vinny development tool (also updated to 32-bit). In addition, a new set of 32-bit example programs have been included on this release.

1.2.1 16-bit and 32-bit Development

For 32-bit development you need the new 32-bit MS-Windows RunTime Revision 5.0 (RTIWIN32.EXE) and Visual Basic 5.0, with Service Pack 3 installed.

Vinny fully supports both 16-bit and 32-bit development. The 16-bit development environment hasn't changed; you still need an NPL Release 4.2x RunTime for Windows (RTIWIN.EXE) and the 16-bit version of Visual Basic 4.0.

With the introduction of NPL Release V, Vinny is now distributed as a standard component of the Niakwa IDE CD-ROM in the following directory.

```
G:\NPLV_IDE\VIM
```

The Niakwa IDE CD-ROM includes a complete set of all Vinny files. For 32-bit development, the following new 32-bit files have been added:

```
VnDev32.bas
VnUtil32.bas
VnLink32.frm
VnChar32.frm
VnplMb32.cls
```

All 16-bit development support files names have remained the same with one new addition.

```
VnplDev.bas
VnplUtil.bas
VnplLink.frm
VnplChar.frm
VnplMb.cls        <=== New File
```

Regardless of which environment you're developing for, both 16-bit and 32-bit development can be done on Windows 95/98 and/or Windows NT.

1.2.2 Instant Vinny

Instant Vinny is now included with Vinny. The installation program will install the main Instant Vinny files (InstVnpl.npl and InstVnpl.bas) in the main VNPL folder and create a sub-folder named IVDemo containing the IV demo programs.

NOTE: To run 32-bit Vinny and Instant Vinny applications, you need to be running all 32-bit components (RTIWIN32 or RTPWIN32, Vinny 5.0, VB5, 32-bit OCX's, and 32-bit DLL's).

NOTE: In order to run 16-bit Vinny and Instant Vinny applications, you need to be running all 16-bit components (RTIWIN or RTPWIN, Vinny 2.10, VB4, 16-bit OCX's, and 16-bit DLL's).

1.2.3 New Demo Programs

There are now 4 different sets of example programs, contained in 4 sub-folders of the main Vinny folder. Each sub-folder contains 2 sub-folders of its own, one for the 16-bit version of the example and another for the 32-bit version, as follows:

```
Demos             HelloC
   Demos16           Hello16
   Demos32           Hello32

IVDemo            HelloE
   Demo16            Hello16
   Demo32            Hello32
```

The main example folders contain the NPL boot program and main diskimage, along with the executables produced by the two VB projects. The VB projects are contained in the two sub-folders of each main example folder.

As the structure implies, there is only one NPL diskimage for each example program. The NPL code used for both the 16-bit and 32-bit halves of an example program are the same. This is not possible with the VB half of the examples because VB 5.0 files are not backward compatible with VB 4.0 files, and some VB 5.0 controls are not backward compatible with VB 4.0 controls. The VB code is, however, 100% compatible between the two versions.

1.3 Installation Notes

Release V of Vinny is shipped on the Niakwa IDE CD-ROM and is installed using the Windows setup program contained on the NIAKWA IDE CD-ROM in the following directory.

```
G:\NPLV_IDE\VIM\SETUP
```

The above assumes G: is the current CD-ROM spec. Adjust accordingly. The setup program will guide you through the rest of the installation. Refer to the Visual NPL Developers Guide for complete documentation on the use of Vinny.

1.3.1 Icons and Paths

The setup program creates a Windows folder named "Visual NPL 5.0". This folder contains 17 icons:

- 8 example VB projects
  - 4 16-bit examples
  - 4 32-bit examples
- 8 corresponding example NPL projects
  - 4 16-bit examples
  - 4 32-bit examples
- 1 README.TXT file

All of the VB icons use the project file name as the target, which means that double-clicking on them will run the last VB you installed. If you have more than one version of Visual Basic on your system then you will have to modify some of the VB icons. You will have to explicitly specify the Visual Basic EXE name and use the project file name as a parameter.

The NPL icons depend on the NIAKWA_RUNTIME environment variable to get the path of the appropriate RunTime. In some cases this may not work, Windows 95/98 may add C: at the beginning of the path, or the environment variable may not be present. If this happens, you will have to explicitly specify the "Target" field for the icons as the full path of the appropriate 16-bit or 32-bit NPL RunTime.

1.3.2 Registering DLL’s and OCX’s

When installing the 16-bit DLLs and OCXs on Windows NT, the setup program may produce an error message saying that the files were not properly registered. This message can be ignored as the files are properly registered later in the install procedure.

1.3.3 Visual Basic 5.0

In order to use the 32-bit version of Vinny, you must have Visual Basic 5.0 installed along with Visual Basic Service Pack 3. The service pack can be downloaded from Microsoft’s FTP site at:

ftp://ftp.microsoft.com/transfer/outgoing/developr

Use your browser and enter the above address. This will display a page of download links in alphabetical order by file name. Download (13MB) and run the file named:

vbsp3.exe

1.4 Upgrading Projects from Previous Releases

1.4.1 Product Changes

This section also pertains to upgrading from Vinny 2.10 to 5.0.9. The developer modifiable modules (NPL-VnplDev, VnplDev.bas and VnDev32.bas) have changed. You should make a backup copy of these modules in each project you intend to upgrade to version 5.0.9.

‘VnMsgBox Function Moved

The 'VnMsgBox function has been moved from the VnplDev module into the Vnpl module. A Visual Basic class module (VnplMb) has been added to correct a focus/ownership bug in 'VnMsgBox when a connection to VB is open. Since this class is referenced by the VnLink form, any applications that include VnLink.bas should now also include the class module VnplMb[32].cls A new function 'VnErrMsgBox has been added to VnplDev to be used for displaying developer modifiable error messages in that module. Its parameters are identical to 'VnMsgBox.

Changes to NPL-VnplDev module

The external function declaration of 'VnMsgBox has been moved into the Vnpl module. A new external function called 'VnErrMsgBox is now declared in VnplDev. The parameters are identical to 'VnMsgBox. The 'VnErrFunc procedure uses this function to display developer modifiable error messages (by default). Simply change all references to 'VnMsgBox to 'VnErrMsgBox.

Changes to VnplDev.bas and VnDev32.bas (same in both)

The new versions of both of these modules should be installed in existing Vnpl projects.

The developer may copy over the new version of VnplDev and then cut and paste those sections that they had modified in the older version (primarily the VnSetObj function, but any of the other functions may also have been modified). The developer may also do it the other way around, keeping the existing version in place and cutting and pasting from the new version (two cut and paste operations totaling 16 lines of code).

1.4.2 Converting Vinny programs from 16-bit to 32-bit

There are 9 steps involved in converting an existing 16-bit Vinny program to a 32-bit Vinny program as follows:

1. First of all, decide whether or not you will maintain the 16-bit version in parallel with the 32-bit version. This means both versions would share the same NPL files but have separate VB project directories and EXE files. The demo programs use this method. This would require separating the NPL files (libraries, boot files, etc.) from the Visual Basic files and moving the VB files into a sub-directory of the NPL project directory. This means you will have to modify your VB project since the location of the VNPL root files will have changed. A second sub-directory would then be created for the 32-bit VB files to go in.

   For example:

   ```
   VNPL                  (contains Vnpl.npl, VnplUtil.bas, VnUtil32.bas, etc.)
   MyProject             (contains NPL files)
       VB16 directory    (contains 16-bit VB files)
       VB32 directory    (contains 32-bit VB files)
   ```

   The NPL code would require minor adjustments in order to connect to the correct VB executable. Consult the demo code to see an example of this.

   If you don't plan to share NPL code between the projects then you can just create a new sub-directory under the VNPL working directory and copy all of the files into it from the existing 16-bit directory.

   For example:

   ```
   VNPL
       16-bit directory  (contains NPL & 16-bit VB files)
       32-bit directory  (contains NPL & 32-bit VB files)
   ```

2. Install the 32-bit versions of the OCX's that are used in your project if they are not already installed. Most of the controls that come with VB4-16 are included in VB5, but third party controls should all be checked.

3. Open the project in VB4-16 and remove VnplDev.bas, VnplUtil.bas, VnplLink.frm, and VnplChar.frm (if it exists) from the project. Then save and close the project.

4. Copy VnDev32.bas from the VNPL directory into the new 32-bit VB directory.

5. Open the project in VB5 and add the VnDev32.bas, VnUtil32.bas, VnLink32.frm, and VnChar32.frm (if required) to the project.

6. Open VnDev32.bas and copy the changes you made to the 16-bit VnplDev.bas. The easiest way to do this is to open VnplDev.bas in another editor (Niakwa Workbench or Notepad) and copy/paste the code via the Windows clipboard. You will most certainly need to update the VnSetObj routine, and you may also need to update the VnCtrlType, VnDevDef, VnErrMsg and VnSetCtrl routines.

7. Save the project. It will now be in VB5 format and you will no longer be able to open it with VB4.

8. Make a new EXE file. If you are sharing NPL code with a 16-bit VB project then you must make sure to name the EXE file something different (such as Demos16.exe and Demos32.exe).

9. Modify or make a new shortcut to run the 32-bit version of your application. You will now be executing RTIWIN32.EXE and loading the VNPL32.DLL library.

1.5 Instant Vinny Changes

Instant Vinny now has two NPL modules, InstVnpl and InstVwin, both of which must exist in the current diskimage. The InstVnpl.npl library contains both of these modules as does the IVDemo.npl library. The method for implementing Instant Vinny remains unchanged. As with Vinny, Instant Vinny now fully supports both 16-bit and 32-bit development.

1.5.1 Instant Vinny Installation

Instant Vinny is now an installed component of Vinny and no longer requires it's own setup routine. Refer to section 1.2 above for details on installing Vinny.

1.5.2 Additions to the InstVnpl Library

New Constants

_VnIVWin32	0 if running RTIWIN, 1 if running RTIWIN32

New Functions

```
FUNCTION 'VnIVMsgBox( /POINTER _Title$,
                      /POINTER _Msg$,
                      Flags)
```

Returns:	Same as 'VnMsgBox

NOTE: Parameters are identical to 'VnMsgBox and the function works the same but can operate on top of Instant Vinny forms. This function existed in the previous version of Instant Vinny but was undocumented.

FUNCTION 'VnIVVersion$

Returns:	The version number of the Instant Vinny library.

PROCEDURE 'VnIVSetNplWndShow(Mode)

This procedure mimics the 'VnSetNplWndShow procedure but disables RTIWIN32 from grabbing focus when it is hidden and resets the focus grabber when it is made visible again. This procedure should not be used in standard Vinny forms nor should it be mixed with the standard VNPL call. It also requires adding a second parameter to the VB VnSetHost procedure for every form that it will be used in conjunction with.

-Example-

```
'VnIVSelect("MainMenu")                    : ;Display an Instant Vinny form     ...

REM The developer should include code here to mimic the display of the NPL window
REM on the Instant Vinny form.  This may involve transferring the NPL screen buffer
REM over to VB and the formatting it in a text box or picture control.

'VnIVSetNplWndShow(_VnHide)

REM The Instant Vinny form should look the same as it did when the NPL window
REM was visible, Now the regular Vinny form may be displayed on top of it.

'VnMethod("VinnyForm.Show","1")    : ;Display a standard Vinny form modally

'VnSleep

'VnIVSetNplWndShow(_VnShow)
```

Changed VB Procedure

Sub VnSetHost(Ctrl As Control, Optional SetHandles)

The second parameter 'SetHandles' has been added. It should be a boolean value and equal to True (use the new constant SetNplWndHandles). This is only needed if you intend to use the NPL 'VnIVSetNplWndShow procedure to hide/show the NPL window with the current form. For example:

```
Private Sub Form_Load()

        VnSetHost RTIWINFrame, SetNplWndHandles

End Sub
```

1.6 Vinny Programming with the Workbench

The latest version of the Workbench incorporates a scheme for representing any character in the format of a hex sequence. That sequence begins with a pipe character (|) and is followed by two hex digits, i.e. |F2. It may only be used as the beginning of a hex sequence. This presents two possible problems with Vinny programs.

First, the default parameter delimiter for ‘VnMethod calls is the | character. The Visual NPL Developer’s Guide (Rev 2.0) shows several examples with the | character used in literal character strings. Developers, however, are advised to use the _VnDelim$ constant (defined in the VnplDev module) instead of hard coding a delimiter into parameter strings.

Second, various Windows components such as the open file common dialog use the | character as a separator for things such as file selection criteria. There are three ways to represent a | character in a string:

1. Use HEX(7C)	`Filter$=”text files (*.txt)” & HEX(7C) & “*.txt”`
2. Use |7C	`Filter$=”text files (*.txt)|7C*.txt”`
3. Use the £ character	`Filter$=”text files (*.txt)£*.txt”`

Code in existing disk images and object files which contain | characters will not be harmed by this feature. When opened in the Workbench the | characters will have been replaced by £ characters and the application will perform as before. It is only the creation of new code that this feature must be carefully observed.

1.7 Documentation

For your convenience the following manuals have been provided as PDF files on the Niakwa IDE CD-ROM in the following directory.

G:\MANUALS\PDF_Files\VIM_Docs

The above directory contains the following documents:

VN2GUIDE.PDF	Visual NPL Developer's Guide (Rev. 2.0)
VN2NOTES.PDF	Visual NPL Release Notes (Rev 2.0)
IV1GUIDE.PDF	Instant Visual NPL Developer's Guide (Rev 1.0)
VIM5ADD.PDF	Visual Information Manager Addendum (Rev. 5.0)

The above PDF files may be viewed with the Adobe Acrobat Reader which is also provided on the Niakwa IDE CD-ROM. Refer to "Quick Start" instructions provided with the Niakwa IDE for details on installing the Adobe Acrobat Reader.