

Q'Nial

Using Q'Nial for Windows

Version 6.3

August 2005

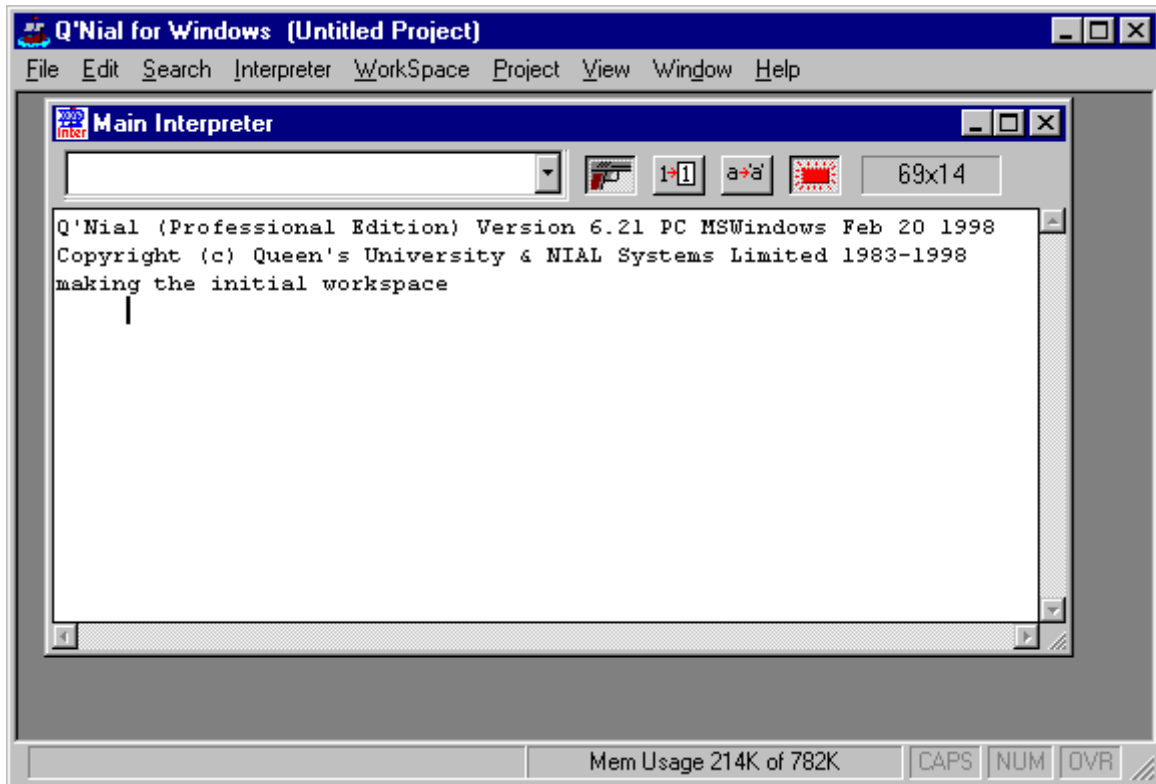
© Nial Systems Limited

Preface

This Manual gives explains the features of the Session Manager provided with Q'Nial for Windows. It begins with a quick introduction to the capabilities and then provides more details on the Session Manager, Interpreter Windows, Editor Windows and Projects.

Chapter 1 Introduction

The Session Manager presents a work area main window with title *Q'Nial for Windows* and an internal window with the title *Main Interpreter*:



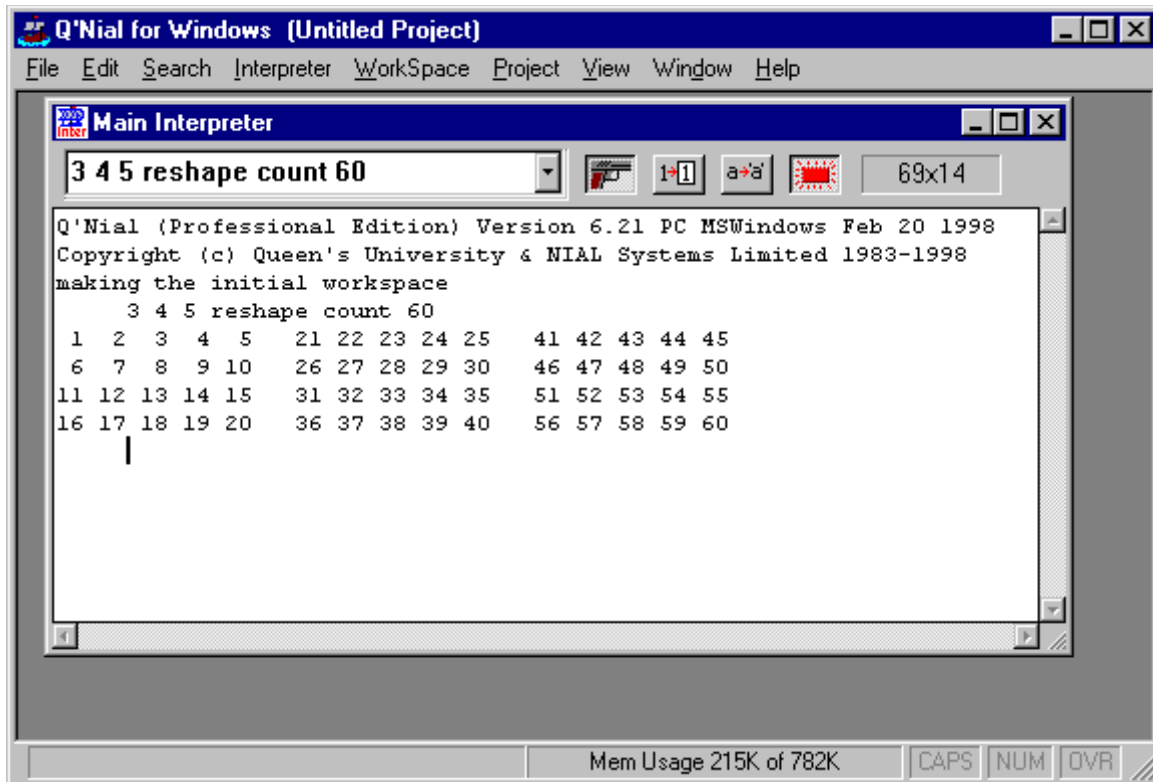
The Interpreter Window provides an enhanced Console style interface to the Q'Nial interpreter. Commands (called actions in Nial) are typed at a prompt of five blank spaces and the results appear as an array picture immediately below at the left margin. Previously typed commands and results scroll off the top of the window and are accessible via the scrollbar.

Previous commands and results can be edited and re-issued as new commands. The most recent commands issued are stored in a pull-down History List for convenience and are also accessible via keyboard shortcuts.

For example, if you type the following command into an Interpreter Window:

```
3 4 5 reshape count 60
```

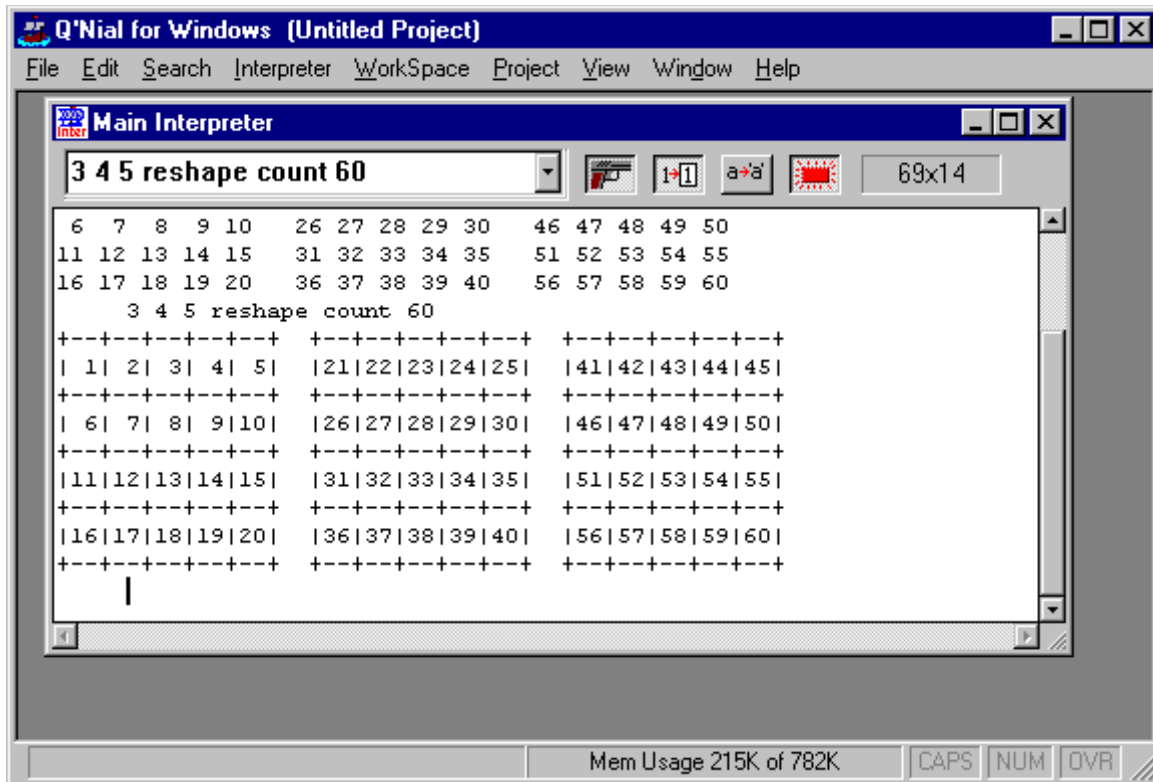
and press the *Enter* key, you will see the results directly following:



The Interpreter Window has a toolbar with a scroll down history window and 4 buttons that control fault triggering, sketch/diagram mode, decor/nodecor mode and messages respectively. It also gives the size of the current window. A sample picture of the toolbar from a different session is:

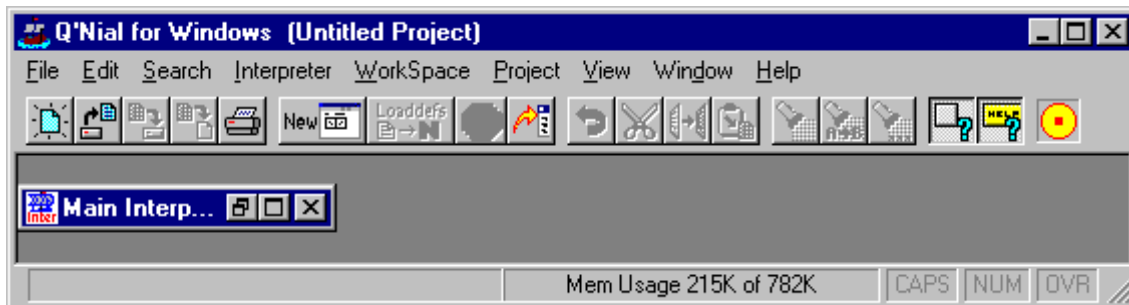


If you click the *sketch/diagram* button on the Interpreter toolbar and then re-issue the same command (by using the key combination of *Ctrl-Enter*, by selecting the same command from the history list, or by moving the cursor up to the command in the Interpreter Window and pressing *Enter* again you will see the following window:



An Interpreter Window provides a convenient interactive code development environment. You can try out small portions of Nial code, and edit then until they work correctly in an interactive and iterative process.

The Session Manager has a horizontal list of menus across the top that change depending on whether the selected internal window is an Interpreter Window or an Editor Window. There is also a Toolbar that is selected by default and controlled by the View|Toolbar menu item.



The contents of Interpreter Window can be printed using the Print button on the Tool Bar, or from the File|Print menu item.

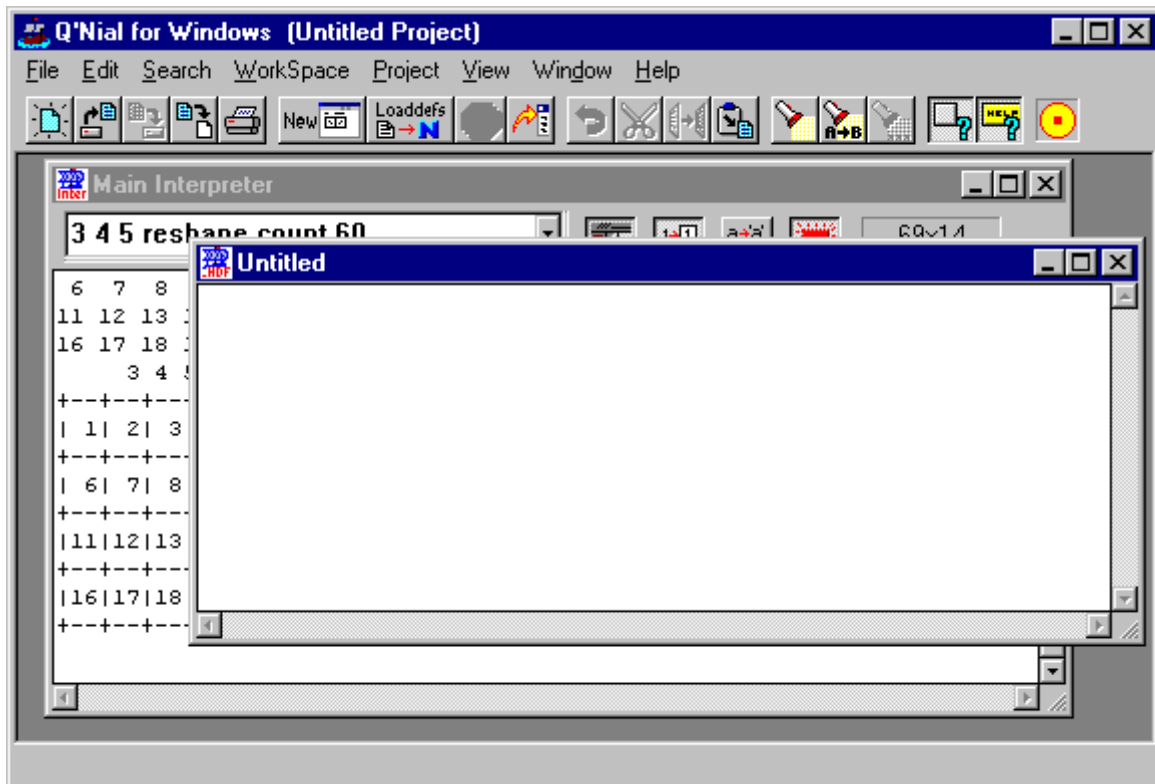
The entire contents of an Interpreter Window (including both issued commands and results) can be logged to a file for later use. To use this feature you set a toggle in the Interpreter Window Properties Dialog. Keeping a log of the Window can be useful because the Interpreter Windows have a limited buffer size. When an Interpreter Window text buffer becomes full, information is thrown away from the top of the buffer.

Editing Nial Definition Files

Most development of Nial program text is done by preparing a text file that contains a sequence of actions. A **Nial definition file**, which has an extension of **.ndf** is a plain ASCII file that contains Nial source code: definitions, statements, comments, expressions, etc. The Session Manager integrates the ability to work with Nial source files in an intuitive manner.

You can open an existing definition file from either the Toolbar or from the File|Open menu item. The file you open is displayed in an Editor Window. An Editor Window has the basic editing capabilities provided by the standard Windows Edit Control used in Notepad and other Windows utilities.

New definition files are created by using the New Button on the Toolbar, or by using the File|New menu item. The file that is created is called *Untitled* until you save it to a specific name.



Interpreting the content of Editor Windows

The content of any Editor Window is passed to the Nial interpreter to be interpreted as a series of actions using the mechanism of the system operation *loaddefs*. This can be done in a number of ways. If an Editor Window currently has the focus, then you may press *Ctrl-L* to issue the *loaddefs* command on the contents of the window, or you may issue the *loaddefs* command in the following ways:

- From the File|Loaddef menu item
- From the “File|Loaddef File” menu item
- From the Popup menu of any Editor Window (right button click)
- From the “Loaddefs” button on the main Tool Bar

The “File|Loaddefs File” menu item prompts you for a definition file name to load. The specified file is loaded into the Nial interpreter with each action segment of the text being interpreted.

All Nial commands generated internally by the Session Manager are processed in the Main Interpreter window. As a result any of the above *loaddefs* methods bring the Main Interpreter window to the top and display the command that was issued.

The content of an Editor Window does not need to be saved to disk for a *loaddefs* command to interpret the content of the Editor Window.

The content of an Editor Window can be printed from the File|Print menu item or from the Print button on the main Tool bar.

All open Editor Windows are saved when you save a Project. The information retained includes the file name and its path relative to the project file, and the size and location of the window. If the content of the Editor Window has not been saved to disk, you are prompted whether you want it saved or not.

Using Projects

This section provides a short overview on the basics of using Projects.

When Q’Nial for Windows starts and no Initial Project has been requested to be loaded, a Default Project is created. The Default Project and any newly created Projects contain only one window. This is the Main Interpreter window. It exists in every Project and cannot be closed. You can then add as many secondary Interpreter Windows and Editor Windows as desired.

You can control what Project is loaded (if any) when the Session Manager starts. This is done through the Startup Setting dialog this is selected by the File|Preferences|Startup Settings menu item. This dialog allows you to specify the name of an initial Project to load, and to check the box indicating that you want it loaded.

If you do not request an initial Project to be loaded, the Session Manager creates a startup Project for you which is untitled. The project can be named and saved to a disk file by using the *Project|Save* or *Project|Save As* menu items.

If you do request an initial Project, then the Session Manager attempts to load that Project. If the specified project file does not exist, the Session Manager creates the default startup Project.

The Startup Settings dialog also lets you specify if and how Projects are automatically saved when they are closed. When a Project is closed, you can request that the Project be saved without asking question, not saved at all with no questions asked, or you can be notified with a prompt that gives you the option of saving the Project or not.

You can create new Projects using the *Project|New* menu item or the similar item on the Popup menu for the Session Manager window background. You are always prompted for a name for a New Project before it is created.

You can save a Project under it’s current name with the *Project|Save* menu item, or under a different name and/or location using the *Project|Save As* menu item. If you have requested that the Project be Auto Saved in the Startup Settings Dialog, then your projects will be automatically saved whenever you change to another Project or quit the Session Manager.

To open an existing Project use the *Project|Open* menu item. The Session Manager requests the name of the Project to open, closes the current Project, and opens the requested Project.

When a Project is save, a Project file is created or rewritten. This file records the state of the session manager including paths to any files open in active Editor Windows. The paths are stored as relative paths to the location of the project file. This approach allows you to move a Project File and the associated definition files from one directory location to another, (on the same computer or to another) and still have the Project work as expected.

The Project Properties dialog allows you to change a number options that affect how the Q’Nial interpreter starts when a project is loaded. Most of the options only take effect when the Project is next loaded. The Dialog has an *Apply and Restart* button that restarts the interpreter with the modified options.

See the section on *Projects* below for more details on Project handling.

Getting Help in Q’Nial for Windows

Q’Nial for Windows has an extensive Help system. The Contents button provides access to a structured overview of the Help topics concerning the Session Manager. The Index button provides a detailed list of all Help topics including help on Nial Language primitives, concepts and syntax.

Context sensitive Help is accessed via the **F1** key almost anywhere in the Session Manager.

You can get Help on dialogs, menus, windows and window areas by simply moving your mouse cursor directly over the item of interest and then pressing the **F1** key. The item of interest does not have to have the input focus or be selected.

Help for Dialog Boxes

Placing you mouse cursor over individual item in a dialog box and pressing **F1** brings up the help for that particular item, whereas placing your mouse cursor over the background area of the dialog box and pressing **F1** supplies general help for the entire dialog.

Contextual Help for the Nial Language

Context sensitive help for Nial can also be accessed using the **F1** key. Selecting a word or placing the edit cursor on a word and pressing the **F1** key displays help on the specified language topic. If a word has been selected, the mouse cursor must remain in the window where the selection occurs while the **F1** key is being pressed.

The help topics on the language are organized into classes such as: concept, syntax, control structures and various groupings of the language primitives. If an exact match for the desired word is found that Help topic is displayed, otherwise the index is displayed starting at the nearest match.

Links in the Help Topics

The Help topics for the language have been divided into classes with the class of each entry displayed at the top. If you click on the word *Class* you go to a list of all classes. If you click on the specific class, the list of topics in that class pops up. You can then click on any topic to bring up its help page. Most Help topics for the Nial language also include *See Also* links that are used to find closely related topics.

There are also many links joining related topics in the Help pages for the Session Manager.

Chapter 2 The Session Manager

Overview

The Session Manager component of Q’Nial for Windows (WinNial.exe) provides an integrated development environment for using the Q’Nial Interpreter under Windows. The Session Manager is based on the standard Windows Multiple Document Interface (MDI) application interface. This provides the Session Manager with a main window that has a desktop area that can contain many other child windows. These child windows cannot move outside the Winnial main window.

This approach allows you to be working on many Nial source files (Editing), and in any number of Interpreter Windows, while keeping them confined to a reasonable area of your Desktop.

The Session Manager supports Interpreter Windows for interacting with the Nial interpreter, and Editor Windows for editing Nial source file and other text files.

The Session Manager also supports the concept of a Project that allows you to manage groups of source files, and different Interpreter Settings for different programming tasks.

Session Manager Main Menu

The Session Manager has a Main Menu that appears in the following form when an Interpreter Window is selected in the Session Manager area:

File Edit Search Interpreter WorkSpace Project View Window Help

The purpose of each of the drop down menus is as follows:

- **File** basic file operations for Editor Windows
- **Edit** standard Edit operations for Interpreter and Editor Windows
- **Search** standard search operations for Interpreter and Editor Windows
- **Interpreter** actions for Interpreter Windows only
- **Workspace** Workspace operations
- **Project** Project operations
- **View** Session Manager Interface object toggles
- **Window** Window arrangement and selection
- **Help** Help for the Session Manager and the Nial language

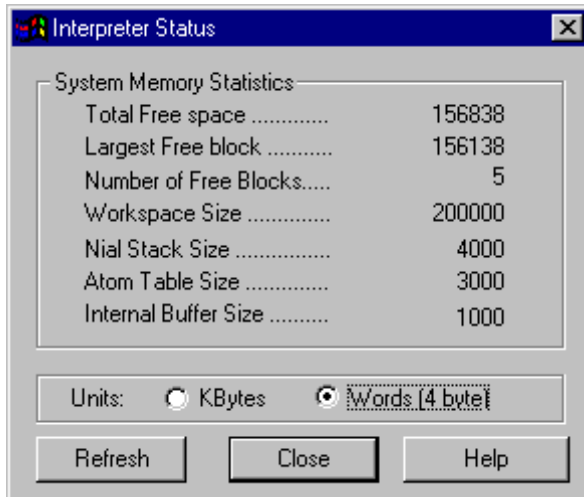
Many of the menu items actions can be initiated from the Tool Bar and also from Popup menus associated with Interpreter Windows, Edit Windows and the background area.

Workspaces Menu

The Session Manager has a Workspace menu to provide convenient access to the workspace management capabilities of the Q’Nial interpreter. This includes access to the Nial commands *load*, *save*, *Clearws*, *Restart* and *Status*. The first two menu items bring up a dialog box for you to provide the filename for the action. The *Status* menu item brings up a Status Dialog described below.



The Workspace Status dialog shows the current memory usage of the Nial Interpreter (not of the entire Session Manager). The statistics are constantly kept up to date while the dialog box is open. This dialog is displayed when the *Workspace|Status* menu item is selected.



The following are the descriptions of the fields:

Total Free Space

This is the amount of memory still available for use by Nial variables and definitions, before the Interpreter attempts to expand the workspace size, or fail.

Largest Free Block

The Nial workspace is divided into blocks of varying sizes. A memory request of this size or less can be satisfied with the current workspace. If this number is small compared to the total free space then it is an indication of memory fragmentation.

Number of Free Blocks

The number of free blocks also indicates the amount of fragmentation in the workspace. A large number here indicates that there are many (probably small) blocks of unallocated memory interleaved between allocated blocks.

Atom Table Size

Phrases and faults are literal atoms used to name variables and definitions, to store symbolic data, and to record error messages. The interpreter maintains a hash table of phrases and faults internally so that they are stored uniquely. The interpreter expands the internal Atom Table as necessary to accommodate a large numbers of phrases.

Workspace Size

This is the current size of the allocated memory reserved for use by the Nial Interpreter. The workspace automatically expands if necessary unless you specifically prohibit it from doing so.

Nial Stack Size

This is the size of the internal Interpreter stack. This expands in size if necessary.

Internal Buffer Size

This is the size of an internal buffer used to store temporary information by the interpreter. This expands in size if necessary.

Units of Measure

The units of measure can be either in words (4 bytes) or in Kbytes.

Status Bar Area

Additionally, in the Status Bar at the bottom right of the Session Manager Window, there is a continuously visible simplified status displayed. It indicates the current workspace size and the amount of the workspace currently in use in Kilobytes.

Normally the Workspace Status Dialog is refreshed after every Nial Interpreter operation. If for some reason the data displayed is not current, the Refresh button will display the current statistics.

Main Toolbar

The Session Manager has the following Toolbar of buttons that are shortcuts to menu items:

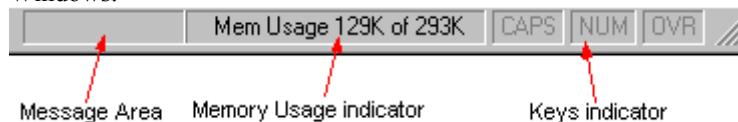


The Main Tool Bar is divided into 6 groups of buttons:

- **File Commands** New, Open, Save, Save As, Print
- **Interpreter Commands** New, Loaddefs, Break Execution, Interpreter to front
- **Editing Commands** Undo, Cut, Copy, Paste
- **Search and Replace** Search , Replace, Repeat
- **Tool Tips** Fly over Hints Toggle, Balloon Hints Toggle)
- **Interpreter Busy Animation** Execution Wheel

Session Manager Status Bar

The status bar is used for displaying Fly over Hints and for display Key states for the Interpreter or Editor Windows.



The Message Area displays Fly over Hints and other messages. The Memory Usage Area indicates the amount of Workspace memory is in use out of the total available.

The remaining sections indicate the state of Caps, Numeric Pad and Insert/Overwrite keys.

Session Manager Startup and Preferences

The Session Manager stores Preferences in the Registry and/or in the winnial.ini file. Preferences for the Session Manager are accessed from the *File|Preferences* menu item. From there, you can set the preferences for:

- Startup
- Interpreter Windows
- Editor Windows
- Projects

The Dialog boxes for Preferences look the same as the Property Dialogs for the same objects, except that the Window title indicates that you are setting Preferences and not Properties. Any changes you make to any of the Preferences as stored in the WINNIAL.INI file. These Preferences are only used when:

- Starting Winnial
- Creating New Interpreter Windows
- Creating New Editor Windows
- Creating New Projects

On the other hand, Properties affect the current object and take effect immediately (except for some Project Properties).

When the Session Manager is started, it either loads an existing Project or creates a new Project for you to start with. If you have specified a default Project to load each time the Session Manager starts, then that Project is loaded. If a Project file is specified on the Winnial command line, then the named Project file overrides the default startup Project.

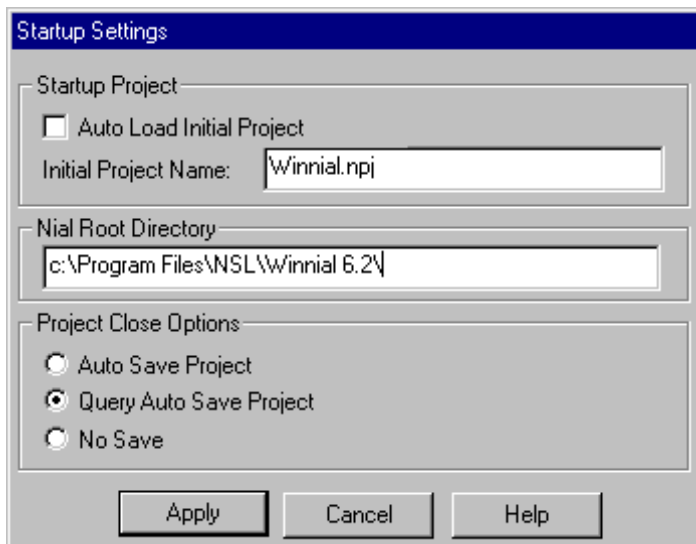
If the user has not requested that a default Project be loaded, the Session Manager starts with a new Project named *Untitled Project*, that contains only the Main Interpreter window.

When the Session Manager is loading a Project, it reads the Interpreter Startup Settings from the Project file to determine how to start the Nial Interpreter. Once the Interpreter is started, all previously opened windows in the project are re-opened and their settings (Fonts, Interpreter Settings, etc.) are restored to the state they were when the Project was saved.

The following Dialog box is presented when you click on the *File|Preferences|Startup Settings* menu item.

Startup Settings

The Startup Settings dialog is:



Startup Project

The Session Manager can be requested to load a specific Project when it starts up. If the *Auto Load Initial Project* option is not selected, the Session Manager starts up with a new project, and requires the user to supply a name for the project when the user saves the project.

Nial Root Directory

This box contains the path to the Nial Root directory. The default is the location determined when Q'Nial for Windows is installed. This can be changed if an alternative Nial Root is desired.

Project Close Options

These options can be used to have the session manager automatically save Projects whenever they are closed, ask if you want the Project saved before saving or not save Projects at all.

Apply Button

Click this button to have the changes take effect.

Exit Dialog

The following Exit Dialog appears if you attempt to **Exit** Q*Nial for Windows when the Interpreter is active:



It provides you with options on whether to terminate or not. You have two choices of exit mechanisms: select one and click on *Continue*; or you can click on *Cancel* to continue execution. If you have turned the Break Execution mechanism off in the Project settings, then you are unable to cause an Exit Dialog to appear.

Attempt to Stop Interpreter First

This choice will try to stop the Interpreter with the Break Execution mechanism. (You may need to do this twice.) If it succeeds, the Interpreter is in a clean state so that if you then Cancel the Exit from the Project Warning dialog, you regain control at the top level.

Forced Exit

This choice forces an Exit without returning to the top level of the Interpreter. If you then Cancel the Exit from the Project Warning dialog, you will return to the execution that was interrupted by the attempt to Exit.

Chapter 3 Interpreter Windows

Overview

The Session Manager may have one or more Interpreter Windows open at any time. While every Interpreter Window shares access to the same Nial workspace, each window may have different Interpreter Settings.

There is always one Interpreter Window open that cannot be closed. This is the *Main Interpreter* Window. It is necessary for the Session Manager to keep this window open at all time so it can receive input and produce output from menu and button commands. Whenever a menu command produces output in the Main Interpreter Window, the window is automatically brought to the top and restored if necessary. For example, a *loaddefs* of the contents of an Editor Window brings it to the top.

You can create an additional interpreter Windows with the Interpreter|New menu item. Each new Interpreter Window is assigned a number, and the window title is *Interpreter N*, where *N* will be the current Interpreter number. You can close the secondary windows as you desires. New Interpreter numbers are set to the current highest Interpreter number plus 1.

Interpreter Button Bar

Each Interpreter Window has a Button Bar at the top of the window:



It provides shortcuts to some features when using Interpreter Windows. The Button Bar contains the following items:

History Drop Down List

The History List is a pull down box that stores the last few commands you have issued in the Interpreter Window. This allows easy visual access to past commands you have issued. You can pull down the box and click on the command you want to have re-issued in the console portion of the Interpreter Window. The text is automatically entered at the bottom of the console area, then it is processed by the Nial Interpreter and the results of the interpretation are displayed (if any).



The History List can also be accessed via the keyboard from the console area. The following key combinations access the History List:

| | |
|-------------------------|--|
| Ctrl-Up | Makes the previous item in the History List the current item. |
| Ctrl-Down | Makes the next item in the History List the current item. |
| Ctrl-Enter | Copies the current item into the console area and executes it and displays the results |
| Ctrl-Shift-Enter | Copies the current item into the console area without executing it. This can allow the user to modify the command. |

The History List does not keep duplicate entries, and once the history list is full (about 50 item), old items are removed from the end of the list.





The History List is saved when a Project is saved and is reset when the Project is re-opened.

Settings Buttons

Immediately to the right of the History List (on the Button Bar), are the Settings Buttons. These four buttons toggle the following Interpreter Settings for the Interpreter Window:

| | |
|-------------------|-----------------------------------|
| Triggering | toggle Fault Triggering on or off |
| Diagram | toggle sketch/diagram mode |
| Decor | toggle decor/noddecor mode |
| Messages | toggle Messages on or off |

These buttons remain synchronized with the state of the interpreter if you issue the equivalent Nial level commands in the console area of the Interpreter Window (or in Nial code executed in definition files):

| Button | Setting ON | Setting OFF |
|---|---------------|---------------|
|  | settrigger l | settrigger o |
|  | set "diagram | set "sketch |
|  | set "decor | set "noddecor |
|  | setmessages l | setmessages o |

Screen Size

The current screen size is always displayed in the right of the tool bar. The size is in characters, and represents the character screen size of the text area. This is useful when adjusting the screen size to accommodate particular output from a Nial program.

Editing in an Interpreter Window

The console area of the Interpreter Window is based on the standard Windows Edit Control. This is where you interact with the Nial Interpreter in a traditional console manner.

The console is a scrolling text edit area that is you can view results, enter and edit commands, and scroll back to see previous results. Typically, you enter Nial commands at the Nial prompt (by default the prompt is 5 blanks), and the results are displayed immediately following the command. Another prompt is issued and you may continue issuing commands. Every command that you issue is entered into the History List.

You are free to move around the console area and *re-edit* previous commands or output. If you then press *Enter*, the modified contents of the line the cursor is on is copied to the bottom of the console area and executed by the Nial Interpreter as if you had typed the command in there directly. This allows you to easily edit mistyped commands, or to develop an expression in an incremental manner. Any changes made to lines above the last line are undone when you press the Enter key so that the Interpreter window provides an accurate history of what you have requested to be executed.

To enable the accurate restoration of the contents of the Interpreter Window, the Delete and Backspace key do not allow you to delete an entire line, but they do allow you to delete characters normally within a line.

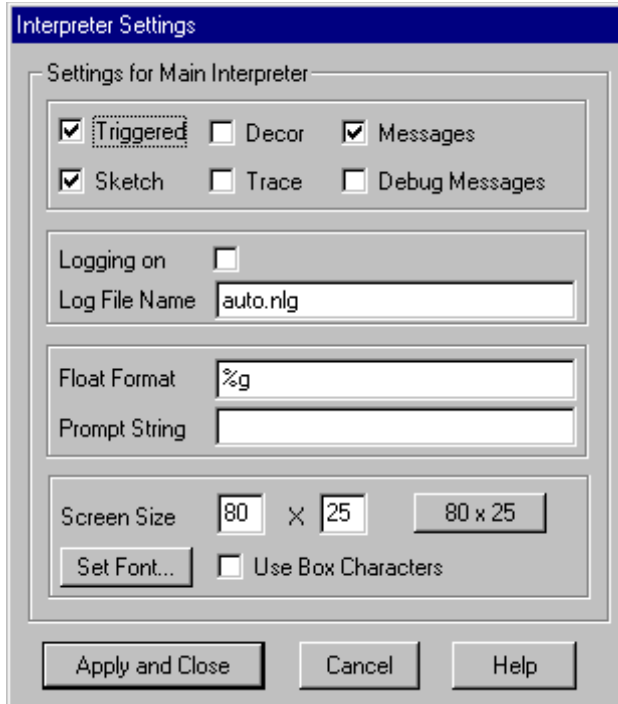
The *Cut*, *Paste* and *Delete* clipboard operations can increase or decrease the total number of lines in the Interpreter Window, and prevent the proper restoration of the contents of the interpreter window. If you use one of these clipboard operations, any changes made in the Interpreter Window up to that point are not restored.

Screen Width

You are free to adjust the size of the Interpreter Window. You can size the Interpreter Window to a specific size using the Interpreter|Properties dialog. The Nial Interpreter is always kept informed of the current screen width (displayed in the Interpreter window Button Bar) so it can produce properly formatted diagrams of Nial arrays. Resizing the Interpreter Window while the Interpreter is busy processing a long running command could produce results that either extend beyond the right edge of the console area, or are smaller than the console area is capable of displaying. The horizontal scroll bar on the console area allows access to results that extend beyond the right edge of the window.

Interpreter Properties

The Interpreter Properties Dialog for any Interpreter Window are accessed in two ways. If the desired Interpreter Window is the active window and you click on the *Interpreter|Properties* menu item then the Properties Dialog is displayed. Alternatively, the Popup menu on the console area on any Interpreter Window has a Properties menu item on it and clicking on it has the same effect.



The Interpreter Properties Dialog can adjust the following Interpreter Settings:

Settings

This section of the dialog allows you to adjust the current settings. Some of these settings are also accessible on the Interpreter Window Button Bar. The following list describes the settings:

| | |
|-----------|---|
| Triggered | Toggles Fault Triggering on or off. |
| Sketch | Toggles Sketch/Diagram mode |
| Decor | Toggles Decor/Nodecor mode |
| Trace | Toggles Trace mode on or off. |
| Messages | Toggles Informative Messages on or off. |

Logging and Log Files

Interpreter Logging is the process of saving all user entered commands and results from an Interpreter Window to a file for later use. By default, Interpreter Logging is turned off. Each Interpreter Window can maintain its own log file and state of logging. The default name for the Main Interpreter log file is *auto.nlg*, and the secondary Interpreter Windows have default log file names of *autoN.nlg* where *N* is the Interpreter Window number.

When logging is turned on, all input and output to the Interpreter Window is saved to the specified log file. Log files are not erased by Q’Nial for Windows when a session terminates, and hence output is appended to the file from session to session. It is up to you to delete log files that you no longer need.

Float (Real Number) Format

In this Text Edit area you enter the C style format specifier (see documentation on *setformat* in the *Nial Dictionary*) that Nial uses to display real numbers. Changing this setting does not affect the internal

precision of floating point number; only the precision of their display. The default is %g.

Prompt String

You can adjust the prompt string that the interpreter issues in the console area of the Interpreter Window. This can be any ASCII string. You might want to designate a particular Interpreter Window to be a debugging console, and thus, you might set it to set the prompt to *DEBUG>*.

It is important to note that, when you press *Enter* in the console area, the Interpreter Windows takes the entire line the cursor is on and eliminates the prompt from the beginning of the line. The remainder of the line is then executed by the Nial Interpreter. If you edit the prompt as displayed in the console area, the Interpreter may not be able to match the prompt properly and the resulting line may not interpret correctly.

Screen Size

You can directly set the screen size of the Interpreter Text Area, or you can use the *80x25* button to conveniently set the size to 80 by 25. Size values are in characters. If the current font is not fixed pitch, then the screen width value is set based on to average of the character widths in the current font.

Fonts

You can select any font for use in the console area of the Interpreter Window. However, it is highly recommended that you use a fixed pitch font because Nial produces output diagrams that assume a fixed pitch font.

Box Characters

Nial is capable of using IBMPC style box drawing characters when creating pictures of Nial arrays for output. However, most fonts do not support these characters. If you are using a font with these characters, then checking this box results in array diagrams with solid lines.

Interpreter Preferences

All of the settings that can be adjusted for an existing Interpreter Window (Properties), can be set as defaults for new Interpreter Windows. The Interpreter Preferences Dialog can be accessed from the *File|Preferences|New Interpreter Settings* menu item.

The Interpreter Preferences Dialog is exactly the same as the Properties Dialog. The difference is that changes made to the Preferences Dialog are saved in the WINNIAL.INI file (or the Registry), and do not affect any open Interpreter Windows. All subsequent Interpreter Windows are created with the new settings in the WINNIAL.INI file.

Interpreter Main Menu

The *Interpreter Window* main menu item is only visible when an Interpreter Window is currently active. It provides the following choices:



| | |
|------------------------|---|
| New | Create a new secondary Interpreter Window. |
| Break Execution | Interrupt the current Interpreter computation. This menu item is only enabled when both the interpreter is busy and the Break Check Project property is set. |
| View Log File | Display the specified log file for the current Interpreter Window in a standard Edit Window. Subsequent updates to the log file (as a result of further interaction with the interpreter window) are not displayed on screen. To view subsequent changes, you must close the log window and then choose this menu item again. |
| Properties | Display the Properties dialog for the current Interpreter window. |

Interpreter Popup Menu

The Interpreter Window Popup menu is activated by Right-Clicking on the editor area of the Interpreter Window.



| | |
|------------------------|---|
| Cut | Cut the currently selected text to the clipboard |
| Copy | Copy the currently selected text to the clipboard |
| Paste | Paste the current clipboard contents to the current cursor position; possibly replacing the currently selected text |
| Font | Change the current screen font. This has the same effect as the Font Button in the Properties Dialog |
| Break Execution | Interrupt the current Interpreter computation. See the Break Check Interpreter Property for more information |
| Clear History | Clear all entries in the pull down history list |
| Properties | Display the Properties dialog for the current Interpreter window. |

Chapter 4 Editor Windows

The Editor Windows provided by the Session Manager are based on the standard Windows Edit Control (effectively the same as Windows 95 Notepad).

Searching

In addition to the basic Editor functionality, the Editor windows have enhanced text searching capabilities. The *Ctrl-S* key combination allows you to do an incremental text search.

Once you press *Ctrl-S* you will see *Search for:* in the message area of the Status bar. Subsequent ASCII keys you press cause a search to occur at each key press. Every ASCII character you press will be added to the search string and displayed in the Status bar message area (i.e. *Search for: "count"*). As you search, matching text is selected and highlighted.

Once you press *Ctrl-S*, you are in *Incremental Search mode*, and you remain in that mode until you press a non-ASCII key, or until the search you are doing fails.

Pressing *Ctrl-S* again, while already in *Incremental Search mode*, searches for the next occurrence of the current search string.

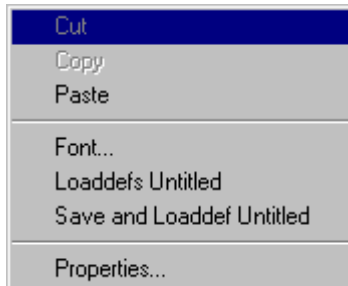
Pressing *Ctrl-S* twice when not in *Incremental Search mode* searches for the last search string (if there is one).

This method of searching interacts with the standard search facilities of Editor Windows and search strings you use in the Incremental mode will be remembered for use when you press F3 (Search again).

This style of text searching is also supported in Interpreter Windows.

Editor Popup Menu

The Popup menu for Interpreter windows is:

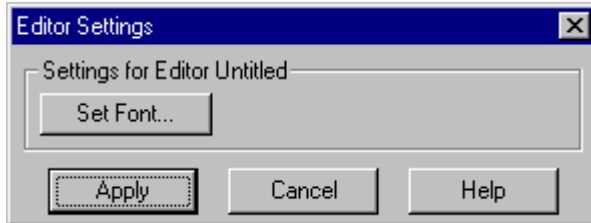


The purpose of each menu item is as follows:

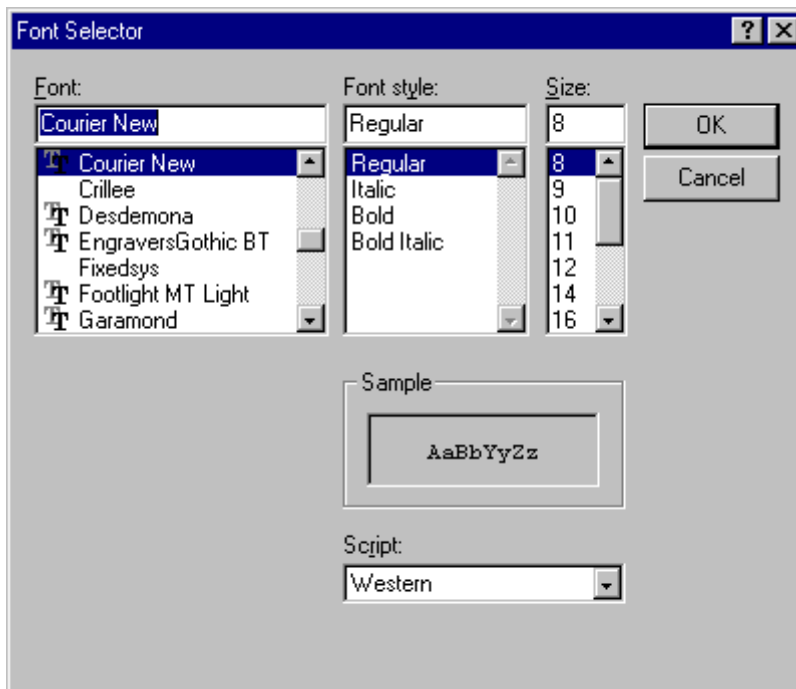
| | |
|------------------------------|---|
| Cut | Cut currently selected text to the clipboard |
| Copy | Copy the currently selected text to the clipboard |
| Paste | Past the contents of the clipboard into the file where the cursor is located |
| Font | Change the font for this Editor window |
| Loaddefs ... | Loaddefs this file into the interpreter |
| Save and Loaddefs ... | Commit the file to disk and then <i>loaddefs</i> this file into the interpreter |
| Properties | Open the Properties Dialog for this Editor window |

Editor Properties

The Editor Properties Dialog for an Editor Window is accessed via the Popup menu in the Editor Window. If you right click on the body of the window then the Popup menu appears. Then you click on the *Properties* menu item and the Editor Properties Dialog is displayed.



The Editor Properties Dialog can adjust the Font setting for the window. It brings up the Font dialog:



which permits you to select a Font, a Font style and a size to be used in the window. If the window is to be used to edit a log file then it is best to choose a fixed sized font in order to make array diagrams understandable.

Editor Preferences

The Font setting that can be adjusted for an existing Editor Window can be set as a default for new Editor Windows. The Preferences Dialog can be accessed from the *File|Preferences|New Editor Settings* menu item.

The Editor Preferences Dialog is exactly the same as the Editor Properties Dialog. The difference is that the Font setting made in the Preferences Dialog is saved in the WINNIAL.INI file (or the Registry), and

does not affect any open Editor Windows. All subsequent Editor Windows are created with the new Font setting in the WINNIAL.INI file.

Chapter 5 Projects

Overview

The Q’Nial Session Manager uses the concept of a Project to help you manage your programming tasks. Projects allow you to save and restore most of the current Session Manager settings. These settings include:

- Locations of all currently open windows
- Interpreter Settings for each of the open windows
- Interpreter Startup Settings
- Workspace to load upon startup and to save on exit
- The directory path to the Project Directory

The Session Manager can start up with a user specified project or with a new project (default). Projects can be auto-saved whenever the project is closed. Projects are saved in binary files with the extension of **.npj**. This is a required extension and the Session Manager enforces that it is used.

Project Basics

When Q’Nial for Windows is run without specifying a Project, or when a New project is created, the project name will be *Untitled Project*. The user can specify an initial or startup Project in two ways:

- By supplying the name of the project file on the command line
- By setting the initial project in the *File|Preferences|Startup* Dialog Box

The first method allows you to create Program Manager or Explorer Icons that startup up specific projects or associations to allow *.npj* file to automatically start Q’Nial for Windows (Winnial.exe). The second method allows the Session Manager to start with a specific project regardless of how or where the Session Manager is started from.

A Project file can also be Dragged and Dropped onto the *Q’Nial for Windows* application and be automatically loaded.

Projects help manage your Nial source code files, Interpreter Windows and Interpreter Settings. The settings saved in a Project file allow each project to automatically load a particular workspace on startup, and to save the same workspace whenever the Project is closed. An initial Nial Definition (*.ndf*) file can also be specified to be loaded when the Project is loaded. Other Interpreter Settings include the ability to specify an initial workspace size, and whether Interpreter execution can be interrupted.

Project Work Area

This is the background area where Interpreter Windows and Editor Windows are displayed. The Windows and their Iconified forms are restricted to the Background area of the Session Manager. The Popup menu for this area is the same as the Projects menu on the Main menu.

New Projects

Newly created Projects start with only the Main Interpreter Window. You are free to create more Interpreter Windows and to open as many Editor Windows as desired. All new projects start with the name *Untitled Project*. When you request a *Save* or *Save As* action for the project you are asked to supply a

name for the project.

The Main Interpreter is created with any Interpreter Preferences that the user has previously set.

Opening a new Project will save the current workspace (if that option is set), then stop the Interpreter, and then restart the Interpreter with the startup options for the new Project.

Projects Saving/Loading

A Project can be saved at any time except when the Interpreter is busy with an execution. Similarly, a Project cannot be loaded when the Interpreter is busy with an execution.

Both saving and loading a Project can involve saving and loading of a particular workspace. If you have specified both an Initial Workspace name and checked that you want it to be loaded when the Project is loaded, then the Interpreter attempts to load that workspace when starting the Project. If the workspace fails to load, the Interpreter starts with a clear workspace.

Additionally, if you have checked that you want the specified workspace to be saved when the project is closed, the Session Manager saves the workspace when the Project closes.

Project Properties

Each Project has an associated set of properties that are saved and restored with the Project. The following properties can be set in the Properties Dialog. This Dialog can be accessed from the *Project|Properties* menu item or from the Popup menu in the background area of the Session Manager.

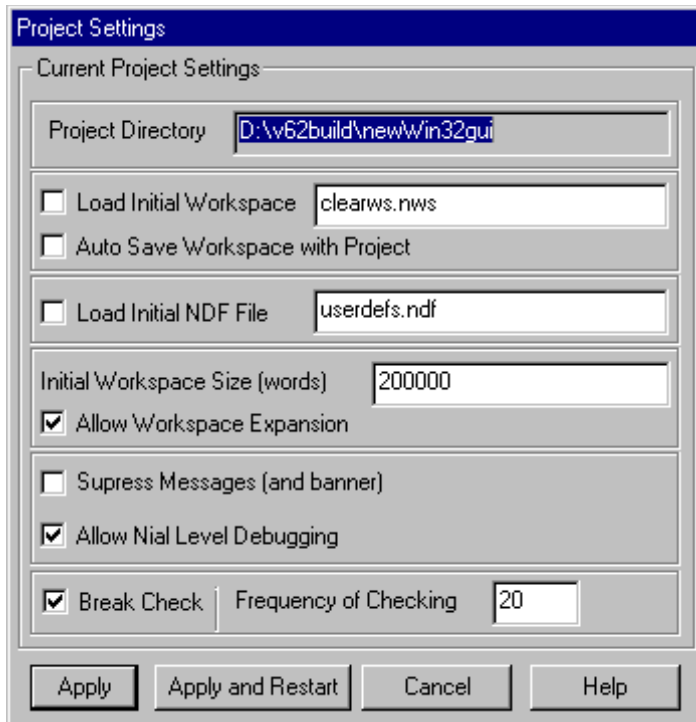
Project Directory

Whenever a Project is created, the project directory is initially set to be the directory where you have specified the project file to be created.

The Project Directory is made the current directory when the Project is loaded. This is done to allow consistent referencing of Nial source files from Interpreter Windows and to reduce the need to type long path names for source files while interacting with the Session Manager.

The Project Directory is displayed in the Project Properties Dialog box, but cannot be changed.

When you attempt to save a Project and there are Editor windows open, the paths to the files being edited are stored in the Project file in relative form from the Project directory. This allows the copying of the Project file and Project Directory and all its contents to a different location. The Project file can then be loaded and all files that were copied are properly located.



Workspaces

Auto Load/Save

By default, the Session Manager creates the initial workspace from internal data unless a *clearws.nws* file exists in the Project Directory, in which case it is loaded. It is possible to specify the name of an initial workspace you wish to load into the Interpreter when a Project is loaded. This can be convenient if you have a set of definitions and variables that you wish to use each time you load a particular project.

It is also possible to request that the workspace be saved with the same name every time the Project is closed. Using this feature preserves any changes in the workspace that occurred in the session, so that you can continue with the Project in the same state at a later time.

Size

The default size of the Interpreter workspace is 200000 words. If you know or expect that you will need a bigger or smaller size, then a different initial size can be set. By default, workspaces can expand transparently if necessary.

Expansion

You can prevent the workspace from expanding from the initial size by using this option.

Initial Definitions

You can specify a Nial definition file to be loaded when a Project is loaded. This is only loaded when the Project is loaded, when a *Project|Restart* is selected from the main menu, or when *Restart* is typed in an Interpreter Window.

Messages


By default the Interpreter outputs informative messages to the active interpreter window (or the Main Interpreter Window if a command was initiated from a menu or button). These messages provide information to you during interactive development. They may inform you of workspace expansions, stack

warnings, and other non-critical Interpreter events. Disable the display of messages if you do not want you Interpreter output to contain these messages.

Debugging

The Interpreter has the ability to disable the Nial level debugging features. This is desirable for production work because the Interpreter runs somewhat faster when debugging is disabled.

Break Checking

Break Checking provides the ability you to interrupt the Nial Interpreter while it is executing. If Break Checking is turned on, you can break execution by using either the *Interpreter|Break Execution* menu item, the *Break Execution* menu item on the Interpreter Window Popup menu, or the Button Bar button .

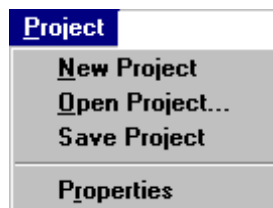
Project Preferences

All of the settings that you can adjusted for an existing Project (Properties), you can also set as defaults for new Projects. You do this using the Preferences Dialog that is accessed from the *File|Preferences|New Project Settings* menu item.

The Project Preferences Dialog has exactly the same form as the Project Properties Dialog. The difference is that changes made to the Preferences Dialog are saved in the WINNIAL.INI (or the Registry), and do not affect the current Project. All Subsequent Projects are created with the settings in the WINNIAL.INI file.

Project Main Menu

The Project Menu allows you to create New Projects, to open existing Projects, or to save the current Project. The Properties dialog for the current Project can also be accessed from this menu.



Create New Project

The current project is closed (and possibly saved), and a new project is created using the Project Preferences. The project name will be *Untitled Project* and a Main Interpreter Window is created.

Open Existing Project

You are prompted for a Project file name for an existing Project. The Project file you have selected is loaded after the current project is closed (and possibly saved if you have set this option in your Startup Settings).

Save Current Project

Selecting this menu item saves all Project information to the current Project file.

Project Properties

This menu item provides an alternate way to activate the Project Properties dialog. Right Clicking on the background area of the Session Manager produces a Popup menu that also has a Properties item on it.

Project Popup Menu

The Project Popup menu is activated by right clicking the background area (not on an Interpreter or Editor Window) of the Session Manager. This menu is the same as the Project main menu.

