NS-DK

Version 5.00 Edition 1

.INI Files

Ref. n° NSLIBU00033

Contents

About this manual

Organization of the manual

This manual contains two chapters, and two appendices.

Chapter 1	**NSLIB.INI File** This chapter presents the NSLIB.INI file which allows the behavior of windows and controls to be configured.
Chapter 2	**NSDKCFG.INI and NSDKLOC.INI Files** This chapter presents the NSDKCFG.INI and NSDKLOC.INI configuration files which replace the former NS-DK.INI file.
Appendix 1	**Labels** This chapter presents various tables about the label redefinition sections associated with graphical objects.
Appendix 2	**Redefinition of keys** This chapter presents various tables about the sections handling redefinition of the actions associated with keys on the keyboard that are configured in the NSLIB.INI file.

Conventions

Typographical conventions

Important term	Important terms are printed in **bold**.
Interface component	The names of windows, dialog boxes, controls, buttons, menus and options are printed in *italics*.
[F9]	Keyboard key names appear in square brackets.
FILENAME	Filenames are printed in UPPERCASE.
`syntax example`	Syntax examples are printed in a `fixed-width font.`

Notational conventions

•	A round bullet is used for lists
♦	A diamond is used for alternatives
1.	Numbers are used to mark the steps in a procedure to be carried out in sequence

definition	A **definition** has a special presentation. It explains the term in a single paragraph. The term appears in the first column, then once in bold in the definition.

Operating conventions

Choose *XXX \ YYY*	This means you need to open the *XXX* menu, then choose the *YYY* command (option) from this menu. You can perform this action using the mouse or mnemonic characters on the keyboard.
Click the *XXX \ YYY* button	This means you need to display the tool bar named *XXX*, then click the *YYY* button in this tool bar (the name of each button is shown by its help bubble). You can only perform this action with the mouse.
Choose the *XXX* button	This means you need to choose the *XXX* button in a dialog box. You can perform this action using the mouse or mnemonic characters on the keyboard.

Icon codes

☞	**Comment,** note, etc.
👓	**Reference** to another part of the documentation
⚠	**Danger**: precaution to be taken, irreversible action, etc.
💡	**Suggestion**: helpful hints, etc.
⇗	**To go a step further**: level of detail or expertise greater than the average level of the document

Chapter 1 NSLIB.INI File

This chapter presents the NSLIB.INI file which allows the behavior of windows and controls to be configured.

You will find in this chapter

- The different sections of the NSLIB.INI file.
- The new sections and parameters.
- Customization of the user interface for porting applications.

Contents

Introduction

The NSLIB.INI file allows the behavior of windows and controls to be configured. The various configurations are grouped by section.

Each of these sections contains the items with their value.

Syntax is as follows:

```
[Section1]
item1=value
item2=value
; Comment

[Section2]
item1=value
```

`[SectionXX]` designates a set of options.

For example:

```
[Fonts]
```

`Item1, Item2, ...` designates the configurable items for the section.

For example:

```
Helv,12=-adobe-helvetica-medium-r-*--*-180-100-
         100-*-*-iso8859-1
```

The sections

There follows a summary of the different sections of the NSLIB.INI file:

- Sections `[NameOfControl.Translations]` are used to change the assignment of the different keyboard shortcuts corresponding to the commands relating to these controls.
- The Section `[Window]`: Behavior of windows when they are being closed and the order of CHECK and TERMINATE events between a parent window and its child windows.
- The section `[Combo]`: Behavior of Combo Boxes and window controls when the [Enter] key is used with these Combo Boxes.
- The section `[Entry]`: Behavior of the selection of an Entry Field when the Entry Field is clicked on.
- The section `[WinFontSubstitutes]`: Character set translation under Windows.
- The section `[Compatibility]`: Ensures compatibility between version 1.01C and versions 1.6x (NS-DK) or 2.6x (NatStar).
- The section `[WinSpecific]`: Configuration specific to Microsoft Windows.

- The section `[ListBox]`: Behavior of List boxes.
- The section `[BitMap]`: Behavior of BitMaps.
- The section `[NS-Trace]`: Not used.
- The section `[Miscellaneous]`: Various settings.
- The section `[Year2000]`: Settings relating to year 2000 problems.
- The section `[Fonts]`: Settings relating to character set problems.
- The section `[FileSelector]`: Behavior of file selectors.
- The section `[DefPalette]`: palette colors.
- The section `[SheetTreeBox]`: Behavior of SheetTreeBox.

An example of an NSLIB.INI FILE

```
; Background character of the entry field
[Entry]
Background=_

; Definition of key sequences of the entry field
[Entry.Translations]
PreviousWord=Gold+Left
NextWord=Gold+Right
ToggleInsert=Insert

; System colors
[ColorPalette]
Disable=Blink

; Initial configuration of the keypad
[Keypad]
Numeric=Normal
Type=Emul3270
```

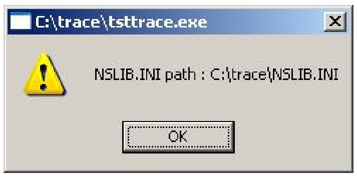
Installation

The NSLIB.INI file is found in the following directories (in order):

1. in the current directory
2. in the directory containing the executable file (allowing an NSLIB.INI file per executable file, therefore)
3. in the directory pointed to by the NS-CFG environment variable
4. in the directory pointed to by the NS-INI environment variable (without the name of the project, therefore)
5. in the directories pointed to by the PATH environment variable.

If no NSLIB.INI file is found, or if one or several sections are missing or empty, default values are applied.

To know the localization of the NSLIB.INI file read by the executable, position the **NS_SHOW_INI** environment variable with YES OR TRUE. Then, a message of information is displayed.


C:\trace\tsttrace.exe

NSLIB.INI path : C:\trace\NSLIB.INI

UNIX

For X Windows targets, the NSLIB.INI file is found in the following directories (in order):

1. The current directory
2. The directory indicated by the NS_CFG (UNIX) environment variable or the NS-CFG (VMS) logical name
3. The SYS$LIBRARY directory (VMS only).

The first NSLIB.INI file found is loaded by the libraries. It is not possible to concatenate the options contained in several files.

For character targets, the NSLIB.INI file is found in the following directories (in order):

1. The current directory
2. The directory indicated by the **NS_CFG** (UNIX) environment variable or the **NS-CFG** logical name under VAX/VMS.

3. SYS$LIBRARY directory under VAX/VMS.

VMS

To define a file common to all users in the SYS$MANAGER directory, add the following command to the procedure SYS$STARTUP_V5.COM :

```
$DEFINE/SYSTEM NS-CFG SYS$MANAGER.
```

An example of an NSLIB.INI file where all the lines are in the form of comments (except translations of the text for localization) is provided with the files making up the character target concerned.

In addition, if one activates the trace mode during the compilation of a project, the sections and the parameters read and used in NSLIB.INI file are indicated in the trace file.

```
NSDK:12:03:19
***************************************************************************
NSDK:12:03:19 NSGENTRACE-exe { Exe : C:\trace\TSTTRACE.EXE}
NSDK:12:03:19
***************************************************************************
NSDK:12:03:19 NSGENTRACE-ini { Nslib.ini file location :  C:\trace\NSLIB.INI }
NSDK:12:03:19 USED TOPICNAMES {
NSDK:12:03:19      [System]
NSDK:12:03:19              RespectWindowSettings=False
NSDK:12:03:19      [Entry.Translations]
NSDK:12:03:19              ToggleInsert=F11
NSDK:12:03:19              ToggleInsert=Insert
NSDK:12:03:19      [List.Translations]
NSDK:12:03:19              PreviousLine=Down
NSDK:12:03:19              NextLine=Up
NSDK:12:03:19      [Frame.Translations]
NSDK:12:03:19              Help=F1
NSDK:12:03:19      [Dialog.Translations]
NSDK:12:03:19              ToggleSelect=F10
NSDK:12:03:20      [Window]
NSDK:12:03:20              HideOnClose=True
NSDK:12:03:20              BroadcastCheck=True
NSDK:12:03:20      [Combo]
NSDK:12:03:20              ExecuteDefPush=True
NSDK:12:03:20              FilterButtonDowns=True
NSDK:12:03:20      [Entry]
NSDK:12:03:20              InsertMode=True
NSDK:12:03:20              MouseAutoSelect=True
NSDK:12:03:20      [WinFontSubstitutes]
NSDK:12:03:20              Helv=MS Sans Serif
NSDK:12:03:20      [PMFontSubstitutes]
NSDK:12:03:20              MS Sans Serif=Helv
NSDK:12:03:20      [PMSpecific]
NSDK:12:03:20              MsgQueueSize=100
NSDK:12:03:20      [WinSpecific]
NSDK:12:03:20              3DDialogs=True
NSDK:12:03:20              UseCtl3D=False
NSDK:12:03:20              MoveWithOwner=True
NSDK:12:03:20              UseLookXp=trueNSDK:12:03:20
IsTabsSelectedAfterInit=True
NSDK:12:03:20      [Entry]
NSDK:12:03:20              InsertMode=True
NSDK:12:03:20              MouseAutoSelect=True
NSDK:12:03:20      [Miscellaneous]
NSDK:12:03:20              Zooming=True
NSDK:12:03:20              BmpZooming=True
NSDK:12:03:20              NoMenuPictures=False
NSDK:12:03:20              NoMenuBarPictures=True
NSDK:12:03:20 } USED TOPICNAMES
NSDK:12:03:20
***************************************************************************
NSDK:12:03:20
***************************************************************************
```

Description of the sections

The NSLIB.INI file is made up of sections, each one comprising a list of parameters with the values specified as being, for example: 'True' or 'False'.

Sections <ControlType>.Translations

The <ControlType>.Translations sections allow previous keyboard shortcuts to be redefined.

For example, by default the Copy action is defined by the two keyboard shortcuts [Ctrl]+[Insert] and [Ctrl]+C. You can remove one of them if you like.

For example, the Paste action is defined by the two keyboard shortcuts [Shift]+[Insert] and [Ctrl]+V. You can remove the one you used the less.

Sections	**This section allows the configuration of keyboard shortcuts associated with ...**
`[Push.Translations]`	push buttons
`[Entry.Translations]`	entry fields
`[MLE.Translations]`	multiple line entry fields.
`[List.Translations]`	List boxes (list fields).
`[Check.Translations]`	check boxes.
`[Radio.Translations]`	radio buttons.

`[Combo.Translations]`	Combo Boxes.
`[ComboEntry.Translations]`	Combo Boxes with an entry field.
`[Frame.Translations]`	window frames (menu, system menus).
`[Menu.Translations]`	menu bars
`[Dialog.Translations]`	dialog boxes

Section [Window]

The section `[Window]` allows the configuration of certain parameters controlling the behavior of windows. Its syntax is as follows:

```
[Window]
HideOnClose=True | False
BroadcastCheck=True | False
```

`HideOnClose`

allows the definition of the behavior of windows when they are being closed:

- If `HideOnClose` is **True**, windows are hidden before being closed.
- If HideOnClose is **False**, windows are not hidden before being closed. Thus, in the case of an MDI window, if one of its child windows returns a CHECK other than 0, the parent window does not need to be re-displayed.

`BroadcastCheck`

allows the order of CHECK and TERMINATE events between a parent window and its child windows to be defined:

- If `BroadcastCheck` is **True**, the CHECK on the parent window is executed first, then the CHECK on the first child window, then the CHECK on the next child window, etc., then the TERMINATE on the first child window, then the TERMINATE on the next child window, etc., then the TERMINATE on the parent window.
- If `BroadcastCheck` is **False**, the CHECK on the parent window is executed first, then the TERMINATE on the parent window, then the CHECK on the first child window and its TERMINATE, then the CHECK on the next child window and its TERMINATE, etc. This operation corresponds to that existing up to version 1.01c of NS-DK.

In the first situation, it is possible to force the closure of child windows at their CHECK event (CLOSE instruction) so that their TERMINATE event may be executed immediately afterwards.

In the second situation, to close child windows before TERMINATE is executed on the parent window, closure should be handled in the CHECK event on the parent window.

Default values

```
[Window]
HideOnClose=True
BroadcastCheck=True
```

Section [Combo]

The section [Combo] allows certain parameters controlling the behavior of Combo Boxes to be configured. Its syntax is as follows:

```
[Combo]
ExecuteDefPush=True | False
FilterButtonDowns=True | False
ExecutedOnLoseFocus=True | False
AlwaySelected=False | False
ChangedOnCloseChild=True | False
```

`ExecuteDefPush`

allows the extent of the effect of the [Enter] key to be defined when it is used in a Combo Box list or a CBE.

- If `ExecuteDefPush` is **True**, pressing the [Enter] key not only executes selection in the list and closure of the list, but also the Push Button associated with this key.
- If ExecuteDefPush is **False**, pressing the [Enter] key only acts on selection in the list and closure of the list.

☞ Associating button / [Enter] key is done in NS-Design in the Return field of the Dialog box info box.

`FilterButtonDowns`

allows definition of whether or not the BUTTONDOWN event should be generated on Combo Boxes and CBEs.

- If `FilterButtonDowns` is **True**, the BUTTONDOWN event is filtered on these objects and is not, therefore, generated.
- If `FilterButtonDowns` is **False**, the BUTTONDOWN event may be generated on Combo Boxes or CBEs, on condition however, that it is added to the list of events relating to these objects.

☞ Whatever the value of `FilterButtonDowns`, the BUTTONDOWN event corresponding to the list drop-down is always generated.

`ExecutedOnLoseFocus`

Allows definition of whether or not the EXECUTED event should be generated on Combo Boxes and CBEs when they are open and lose the focus.

`AlwaySelected`

Allows definition of whether or not the EXECUTED event should be generated on Combo Boxes and CBEs when an item already in the entry field is selected.

`ChangedOnCloseChild`

Allows definition of the events received if the Entry Field value is changed during execution of the SELECTED event.

- If `ChangedOnCloseChild` is **False** then the events received will be:

  CHANGED
  SELECTED
  CHANGED
  EXECUTED

- If `ChangedOnCloseChild` is **True** then the events received will be:

  CHANGED
  SELECTED
  CHANGED
  CHANGED (with restoration of the old value)
  EXECUTED

Default values

```
[Combo]
ExecuteDefPush=True
FilterButtonDowns=True
ExecutedOnLoseFocus=True
AlwaySelected=False
ChangedOnCloseChild=True
```

Section [Entry]

This section allows certain parameters controlling the behavior of Entry Fields to be configured. Its syntax is as follows:

```
   [Entry]
   InsertMode=True | False
   MouseAutoSelect=True | False
```

`InsertMode`

allows definition of insert mode by default for Entry Fields and MLEs.

- If `InsertMode` is **True**, the default mode is insert mode and the cursor is represented by a thin vertical bar.
- If `InsertMode` is **False**, the default mode is overtype mode and the cursor is represented by a character sized box.

`MouseAutoSelect`

allows definition of whether or not an Entry Field which did not have the focus should be selected when the mouse is clicked on it.

- If `MouseAutoSelect` is **True**, clicking the mouse on an entry field selects it. To insert characters into the field without deleting the contents, it is necessary therefore, to click a second time.
- If `MouseAutoSelect` is **False**, clicking the mouse on an entry field puts the cursor into insert mode; characters can then be entered immediately without the contents being deleted.

Default values

```
[Entry]
InsertMode=True
MouseAutoSelect=True
```

Section [WinFontSubstitutes]

This section is used to replace an application's character sets chosen by the developer with others that are more suitable or better supported under Windows.

For example, if a NatStar application is ported from OS/2 PM to Windows and the « Helv » font has been used, it should be replaced by a similar font.

```
Helv=MS Sans Serif
```

All displays in the application with the « Helv » font will be replaced with the « MS Sans Serif » font of the same size.

Section [Compatibility]

This option allows compatibility to be ensured between version 1.01C and versions 1.x and 2.x.

```
Compat_101c=True | False
```

When this option is set to True, all the following options are put in 1.01C. compatibility mode.
Each option may however, be changed individually, independently of the general option.

Section	Parameter	1.01C compatible value
[WinSpecific]	EnableByteAlign	False
[WinSpecific]	RoundAveCharWidth	False
[WinSpecific]	AutomaticClipSibling	False
[BitMap]	PointerFocus	False
[Listbox]	NLSSort	False
[Miscellaneous]	CopyStringNegIndex	True
[Miscellaneous]	Pos01c	True
[Miscellaneous]	ConstantClientSize	False
[Miscellaneous]	LongCString	False
[Miscellaneous]	ShowWndForegr	True
[Miscellaneous]	DeferTemplateDllUnload	False
[Combo]	ExecutedOnLoseFocus	True
[Combo]	AlwaySelected	False

Default values

```
 [Compatibility]
Compat_101c=False

[WinSpecific]
EnableByteAlign=True
RoundAveCharWidth=True
AutomaticClipSibling=True

[listbox]
NLSSort=True

[BitMap]
PointerFocus=True
Transparent=False
AutoButtonFrame=False

[Miscellaneous]
Pos01c=False
CopyStringNegIndex=False
ConstantClientSize=True
LongCString=True
ShowWndForegr=False
DeferTemplateDllUnload=True

[Combo]
ExecutedOnLoseFocus=True
AlwaySelected=False
```

Section [WinSpecific]

`3Ddialogs= True | False`

This option is only valid in old look mode (ie. by copying the DLL NSxxWPS3.DLL under the name NSxxWPS.DLL). It acts on the « 3D » borders of dialog boxes.

`UseCtl3D= True | False`

This option is only valid under Windows 3.1 and triggers loading of the CTL3DV2 DLL if it exists. This DLL is then used for displaying controls.

`MoveWithOwner=True | False`

This option sets or does not set the mode of movement of secondary windows if the parent window is moved.

`MDIChild_ParentDefaultIcon=True | False`

This option sets the MDI child windows with the same icon than the MDI parent window when the *Default icon* propriety of the child window is ticked. To activate this option, uncomment the line and set it to true.

`EnableByteAlign= True | False`

In version 1.01C, the 'Enable Byte Align' style of windows was never applied under Windows.

`RoundAveCharWidth= True | False`

Uses the 1.01c version algorithm to calculate the size and position of windows according to the size of the system fonts. This algorithm has changed in versions 1.5x.

`DlgBackColorWindow= True | False`

This option is only valid in new look mode (ie. by copying the DLL NSxxWPS5.DLL under the name NSxxWPS.DLL).

By default in this mode, the dialog box background color is light gray. The fact of setting this option to **True** allows this color to be forced to white.

`printBlockingMode=True | False`

Under Windows 95, the REP_START function does not block completely and controls may still be enabled during printing. A mode for blocking printing has therefore been added.

If `printBlockingMode` is **True** windows can no longer receive messages during execution of the REP START function.

If `printBlockingMode` is **False** windows can receive messages during execution of the REP START function.

`DrawClientEdge=True | False`

This option allows windows to be drawn without the internal border which gives an aspect of shadow.

`AutomaticClipSibling=True | False`

In version 1.6x (NS-DK) or 2.6x (NatStar), a dialog box and its controls have the attribute CLIPSIBLING when a child box is not a control (another dialog box for example). In version 1.01C the dialog box and its controls do not have the CLIPSIBLING attribute.

`UseLookXP= True | False`

This is used to apply the Windows XP or Vista look to the user interface. When set to **False**, this option then applies a Windows 2000 look to applications using the Nat System runtime. At **True**, Windows outlines the design of the controls itself. This is truly a Windows look, but we cannot totally control what is shown in terms of colour and texture. If the colours and texture of the application’s controls are relevant to user information, this flag should be set to **False**.

`UseClipChildren= True | False`

If anchors are being used in a resizable dialogue box and some of its controls are coloured badly (such as Custom Controls), the problem can be corrected by the setting useClipChildren=False. This option will affect the entire application; unlike the CLIENT.CLIPCHILDREN dynamic parameter, which affects the CLIENT pseudo-control of a window.

`UseDevBmp = True | False`

Allows a workaround for a bug in the production of transparency masks with GIF or JPEG images on Windows 95. This option is set to **True** by default on Windows 95, and to **False** on any other platform.

`IgnoreTaskList= True | False`

Allows the main window of an application to be obtained from the list of system tasks ([Alt]+[Tab]), by setting this option to **False**. This condition is necessary but you should also: 1) uncheck the **Task List** check box in the main window **Info** box and in all linked windows, 2) uncheck the **Title bar** check box in the main window **Info** box to avoid having a corresponding icon in the task list.

For main and not secondary windows (windows attached to the desktop), all these windows appear in the task list, except for windows without a title and for which the **Task list** option is not checked. If **IgnoreTaskList=True** is set, these windows always appear in the task list.
For modal or secondary windows:

- If the **Task list** property of the window's properties pane is checked, the window appears in the task list.
- If **IgnoreTaskList=True** is set, these windows never appear in the task list.

To summarize, enabling the **IgnoreTaskList** option means that all the child desktop windows are visible in the Windows task bar and hides the others.

```
VerifyData =True | False
```

Allows detection of the state of the window to avoid sending Windows messages to a closed window.

```
x_translation =number
```

Set X translation to override desktop windows positions. When a window displays, its position is defined in NSDESIGN or that it is modified by order CHANGE or SETPOS, the specified values are added with the initial values.

```
y_translation =number
```

Set Y translation to override desktop windows positions. When a window displays, its position is defined in NSDESIGN or that it is modified by order CHANGE or SETPOS, the specified values are added with the initial values.

Example of use: an application was made with windows of 800/600 size be carried out in 800/600 screens. The windows are designed to position into 0/0. On a 800/600 screen windows occupy all the screen. If one passes on a 1024/768 screen windows will position in the left bottom. If you want to set them on the center of the screen, you will have to set x_translation value at 112 and y_Translation at 84.

(1024-800) / 2 = 112

(768-600) / 2 = 84

Default values

```
[WinSpecific]
3DDialogs=True
UseCtl3D=False
UseClipChildren=True
MDIChild_ParentDefaultIcon=False
MoveWithOwner=True
EnableByteAlign=True
RoundAveCharWidth=True
DlgBackColorWindow=False
printBlockingMode=False
DrawClientEdge=True
UseLookXP= True
UseDevBmp = True | False ;depending on the platforms
IgnoreTaskList=True
```

```
VerifyData=False
x_translation =0
y_translation =0
```

Section [ListBox]

```
NLSSort=True | False
```

From versions 1.6x (NS-DK) or 2.6x (NatStar), STRING comparisons in sorted List Boxes are made taking locales into account. Accents and the difference between upper and lower case are not taken into account. If the `NLSSort` flag is set to False (version 1.01C) comparisons are only made using ASCII or ANSI tables.

```
FocusWidth=1
```

In multiple selection List Boxes, when several lines are selected, it is sometimes difficult to see which has the focus. This parameter allows the number of pixels appearing in inverse video to be changed and so display which line has the focus more easily.

```
ExtendedSelection =True | False
```

This option allows the desired type of selection to be chosen for multiple selection List Boxes.
If `ExtendedSelection` is True, multiple selection is comparable to that of Windows. To select several lines, the [CTRL] key should be kept depressed, to make a selection of adjacent lines, the [SHIFT] key should be kept depressed.
If `ExtendedSelection` is False multiple selection is unchanged; several lines may be selected without pressing the [CTRL] key.

Default value

```
[ListBox]
NLSSort=True
FocusWidth=1
ExtendedSelection=False
```

Section [BitMap]

```
PointerFocus= True | False
```

This option determines if a PushButton type BitMap takes the focus or not when it is clicked on.
In version 1.01C, a bitmap does not take the focus when it is clicked on, from versions 1.6x (NS-DK) or 2.6x (NatStar), it does take the focus.

```
Transparent= True | False
```

If `Transparent` is **False**, the BitMap controls whose dynamic parameter FORECOLOR is set to the default value (TRANSP_DEFAULT%) behave as if FORECOLOR had the value TRANSP_NEVER% (opaque).

If `Transparent` is **True**, the BitMap controls whose dynamic parameter FORECOLOR is set to the default value (TRANSP_DEFAULT%) behave as if FORECOLOR had the value TRANSP_TOPLEFT% (pixels of the same color as that of the top left pixel of the bitmap take the background color of the dialog box or of the dynamic parameter BACKCOLOR of the control if its value is not -1, achieving a transparency effect).
The option `Transparent` may also be configured separately for each application by means of the functions GETBITMAPCONTROLSDEFAULTS% and SETBITMAPCONTROLSDEFAULTS combined with the constant DEFBMPTRANSPARENT%.

```
AutoButtonFrame= True | False
```

When this option is set to **True**, PushButton type BitMap controls only having a Released bitmap (or even having the same bitmap handle as that of the Released bitmap for the Pressed and/or Disabled bitmaps) automatically draw the contours of the released or pressed button as well as the Disabled effect (by withdrawing half the pixels).
The colors used to draw the contours and the background (around the bitmap) are the same as those of a normal Push Button. It is advisable therefore to use this option in combination with the `Transparent` option. In this situation, the default replacement color (if the dynamic parameter BACKCOLOR of the BitMap control is equal to the default value of -1) is not that of the background of the dialogue box but that of the Push Buttons.
The option `AutoButtonFrame` may also be configured separately for each application by means of the functions GETBITMAPCONTROLSDEFAULTS% and SETBITMAPCONTROLSDEFAULTS combined with the constant DEFBMPBUTTONIZED%.

Default value

```
[BitMap]
pointerFocus=True
Transparent=False
AutoButtonFrame=False
```

Section [Miscellaneous]

Reminder on multi-resolution mode.

Under Windows, the user has a choice between « large font » and « small font ».

Under OS/2 the option « large font » is automatically selected via 1024 x 768 resolution.

This change in size of the font is accompanied by an enlargement of the windows and controls. For more details, see the section « General issues to do with development in multi-resolution mode » in the NSMISC library.

`Zooming=True | False`

This parameter allows definition of the type of window that is going to be automatically enlarged.

If `Zooming` is **False**, only dialog boxes will be enlarged.

If `Zooming` is **True**, all windows of whatever type will be enlarged.

`BmpZooming=True | False`

This parameter enables definition of whether bitmaps not having the « Adjust Size » style will be affected by the change in size.

If `BmpZooming` is **False**, bitmaps will never be enlarged.

If `BmpZooming` is **True**, bitmaps not having the « Adjust Size » style will be enlarged.

`RestrictSkip=True | False`

This parameter allows the behavior of the NCL SKIP function to be changed.

If `RestrictSkip` is **True**, the characters deleted by this function will be limited to: TAB (code ASCII 9), LF (code ASCII 10), CR (code ASCII 13) and SPACE (code ASCII 32).

If `RestrictSkip` is **False**, all the characters whose ASCII code is less than or equal to 32 will be deleted by this function.

`CopyStringNegIndex=True | False`

In version 1.01C, a copy of a character string with a negative index, for example A$ (1..-2), returns the whole string. From versions 1.6x (NS-DK) or 2.6x (NatStar) it returns an empty string.

`Pos01c=True | False`

When it is passed an empty character string as the first parameter, the POS function returns 1 in version 1.01C but 0 from versions 1.6x (NS-DK) or 2.6x (NatStar).

`ConstantClientSize=True | False`

In version 1.01C, the size of windows is constant while from versions 1.6x (NS-DK) or 2.6x (NatStar), it is the size of the client part of the window that remains constant.

`LongCString= True | False`

From versions 1.6x (NS-DK) or 2.6x (NatStar), CSTRINGs longer than 255 characters are supported. This is not the case in version 1.01C.

`ShowWndForegr= True | False`

In version 1.01C a Show on a window causes it to appear in the foreground, something which is not the case from versions 1.6x (NS-DK) or 2.6x (NatStar).

`AlignNonAutoStaticText= True | False`

Modifies the vertical alignment of Static Texts which don't have the AUTOSIZE attribute or the WordWrap attribute.

`DeferTemplateDllUnload= True | False`

From versions 1.6x (NS-DK) or 2.6x (NatStar), unloading of DLLS containing templates is delayed during closure of the window containing the template. In version 1.0C it happens instantaneously.

```
NoMenuPictures= True | False
```

☞ New parameter in NatStar.
True allows menu items to be displayed in the left-hand column without their associated images. This allows the use of automatic test tools that need to read the text of application menu items. This is not possible when portable libraries draw the menus to display the images (in this case, Windows APIs for handling menus are not capable of retrieving the text).

```
NoMenuBarPictures=True | False
```

Set to True to allow Unicode windows caption in UTF-8 charset under WinNT/2000/XP.Also important for menu items text when NoMenuPictures=True.

```
MenuOrToolbarPictures= Toolbar | Menu | Both
```

☞ New parameter in versions 1.6x (NS-DK) and 2.6x (NatStar).
The option `MenuOrToolbarPictures` allows the choice of default use of images in the menus of applications generated by NatStar (PM menus) when the option indicating whether the image should appear as a button on a tool bar, beside the corresponding menu item, or both, has not been defined (old applications). The possible values for this option are `Toolbar` (default value, compatible with old behavior: tool bar images only), `Menu` (images in the menu only; the tool bar will be empty if no item is using an explicit option to put buttons on it) or `Both` (menu and tool bar).

1. The tool bar may become empty if MenuOrToolbarPictures=Menu si set. In this situation, if you want to delete it, you must change the call to MNU_OPEN and remove the option MENU_SIMPLE_STATUSBAR%. For further details, see MNU_OPEN.

2. From NatStar 2.61, a new instruction MNU_SETDEFPICTURESUSAGE (in NSCUST.NCL) allows the choice shown in the NSLIB.INI file to be overridden. The options are MDPU_TOOLBAR%, MDPU_MENU% and MDPU_BOTH% which are respectively equivalent to ToolBar, Menu and Both in the NSLIB.INI file.

```
UnicodeWndClass=False | True
```

Allows you to set the window titles in Unicode UTF-8 format under WinNT/2000/XP. It is also used for menu text elements when the parameter `NoMenuPictures=True`.

```
NsWPForceNatNumberSep= True | False
```

Allows you to modify the character to be used as the decimal or the thousands separator. At the start of the application, call SETDECIMALSEPARATOR « , » and SETTHOUSANDSSEPARATOR « . » instructions.

```
HiddenOwnerWindowsDefaulted= True | False
```

Set to false to solve problems with secondary Windows focus, opened when the owner is not yet visible.

```
PasteMoveCursor= True | False
```

So far the 'copy paste' in our products positioned the cursor at the start of the copied string, this was annoying in the case of a continuous input from the keyboard.
To fix this issue Nat System introduced the parameter PasteMoveCursor.
Uncomment this line if you want to position the cursor after the pasted string.
;PasteMoveCursor=True

Default values

```
[Miscellaneous]
Zooming=False
BmpZooming=False
RestrictSkip=False
CopyStringNegIndex=False
Pos01c=FALSE
ConstantClientSize=True
LongCString=True
ShowWndForegr=False
RestrictSkip=False
DeferTemplateDllUnload=True
AlignNonAutoStaticText=FALSE
NoMenuPictures=False
MenuOrToolbarPictures=Both
UnicodeWndClass=False
NsWPForceNatNumberSep=False
HiddenOwnerWindowsDefaulted= True
PasteMoveCursor= False
```

Section [Year2000]

```
Force4Digits=True | False
```

This configuration changes the behavior of date type Entry Fields where the year needs to be entered with 4 figures. (mm/dd/yyyy)
If `Force4Digits` is **False**, 2 figure entry for the year is possible and the current century will be added automatically to the year of the date entered.
If `Force4Digits` is **True**, 2 figure entry is not possible and 4 figures must be entered for the year.

```
MilestoneYear=30
```

For any two figure date, if the year is less than the milestone year then the year is after the year 2000, otherwise, the year is considered as being before the year 2000.
The value of the milestone year is read at initialization. If no value is defined in the file, the default value will be -1. This value of -1, if SETMILESTONEYEAR is not set to change it, will be such that NS-DK and NatStar will behave exactly the same as before, ie. to fill in a 2 figure year, the current century will be taken.

Default values

```
[Year2000]
Force4Digits=False
MilestoneYear=-1
```

Section [Fonts]

```
DefaultVectSizes=6,8,9,10,12,14,18,24,30,36,48,60,72
```

This configuration allows the list of available sizes of TRUE-TYPE fonts to be changed under Windows.
The 13 values offered are the 13 values available as standard.
Changing these values allows other sizes to be chosen.

Example

```
DefaultVectSizes=6,7,8,9,10,11,12,13,14,18,24,30,36
```

☞ This change will be enabled equally in NatStar, NS-DK or in generated applications.

The list of sizes **must** be made up of 13 different sizes.

Default values

```
[Fonts]
DefaultVectSizes=6,8,9,10,12,14,18,24,30,36,48,60,72
```

Section [FileSelector]

The `[FileSelector]` section allows the appearance of NS-DK or NatStar file selectors to be chosen. The colors are the integer numbers of the constants COL_*%. See also the functions GETPATHNAME$ and GETPATHNAMEWITHRELIEF$.

```
BackColor
```

The colors `BackColor` correspond to the background colors of the dialog box.

```
BackColorEntry
```

The background color of entry fields and lists.

```
ForeColorEntry
```

Text color of entry fields and lists.

```
BackColorStatic
```

The background color of headings (by default, `BackColorStatic` takes the same value as `BackColor`).

```
ForeColorStatic
```

Headings text color.

```
Font
```

Allows the name and size of the character set to be defined (for all controls).

```
ShowVolumeNames
```

Allows the list of disks to be indicated for which you wish to display the name of the volume (because it may be long for some types of drive unit); the corresponding letters may be defined explicitly and/or sequences may be given with the first and last letters separated by a minus sign (-).

```
NoStaticRelief
```

Allows the shadow of file selector titles to be removed.

Default values

```
[FileSelector]
BackColor1=27
BackColor2=27
BackColorEntry1=16
BackColorEntry2=16
ForeColorEntry1=17
ForeColorEntry2=17
BackColorStatic1=16
BackColorStatic2=16
ForeColorStatic1=35
ForeColorStatic2=35
Font=Arial,8
ShowVolumeNames=C-Z
NoStaticRelief=False
```

Section [ColorPalette]

The `[ColorPalette]` section redefines the default colors for different parts of windows and controls.

Under Windows, when a user modifies a windows' theme, the theme colors are assigned. By using the `[ColorPalette]` section, you can restore the original colors of your application.

The changes in the color palette do not affect the Windows XP Look & Feel which determines its colors except for dialogs and windows background. The palette changes affect the old Windows Look & Feel under XP as well as all previous Windows versions.

For a full list of standard colors and system colors under Unix, see Appendix B in the *Porting Applications* manual.

`CbackGround`

Background color of WINDOW type windows.

`Neutral`

Colors of control’s text: EntryField, ComboBox, ComboBoxEdit, ListBox et MLE.

`ButtonLight`

The lightest shadow color for CheckBox and Radio Button controls and the lightest color of the disabled text.

`ButtonMiddle`

Inside color of disabled controls (Entry Field, Listbox, CheckBox, Radio Button).

`ButtonDark`

The darkest shadow color for some controls (CheckBox, Radio Button) and the lighter color of the disabled text.

`ButtonDefault`

Push buttons’ border color.

`DialogBackground`

Background color of dialog boxes.

```
HiliteForeground
```

Color of the selected text (Entry Field, ListBox, MLE).

```
WindowStaticText
```

Color of the enabled Static text.

```
ScrollBar
```

The standalone Scroll Bar control background color.

```
Menu
```

Menu items’ background color.

```
Window
```

Light shadow Radio Button and Check Box background color.

```
WindowFrame
```

Frame color of Group Boxes with no shadow.

```
MenuText
```

Menu items’ text color.

```
WindowText
```

Color of some of the portable librairies dialog boxes like the one that can be called from MenuItem *Windows/More* in a MDI application

Default values

```
[ColorPalette]
CbackGround =White
Neutral=Black
;False=LightGray
;True=Black
ButtonLight=Black
ButtonMiddle=DarkGray
ButtonDark=White
ButtonDefault=Black
;Shadow=Black
;IconText=Black
DialogBackground=LightGray
HiliteForeground=White
HiliteBackground=Black
;InactiveTitleTextBgnd=White
;ActiveTitleTextBgnd=Black
;InactiveTitleText=Black
;ActiveTitleText=White
;OutputText=Black
WindowStaticText=Black
ScrollBar=LightGray
;WindowBackground=blue
```

```
;ActiveTitle=LightRed
;InactiveTitle=LightGray
Menu=LightGray
Window=White
WindowFrame=Black
MenuText=Black
WindowText=Black
;TitleText=Black
;ActiveBorder=Yellow
;InactiveBorder=Black
;AppWorkspace=White
;HelpBackground=White
;Tilde=Blue
```

Section [DefPalette]

The `[DefPalette]` section allows the initial 18 colors of the palette to be set (for all the applications generated by NatStar or NS-DK) in the NSLIB.INI file in the section `[DefPalette]`.

Each color is a 32-bit integer whose low order byte contains the blue component (from 0 to 255 or from $0 to $FF), then by the increasing order of the bytes, the green and red components, the high order byte not being used.

☞ The color may be defined as three integers from 0 to 255 (or from $0 to $FF) separated by commas (spaces and tabs on either side of the commas are ignored) or as a single integer from $0 to $FFFFFF.

An example of the section [DefPalette] from the file nslib.ini. Representation of the colors in hexadecimal figures:

```
[DefPalette]
Color1=$rrggbb
Color2=$rr,$gg,$bb
...
Color18=$rrbbgg
```

The letters rr, gg and bb represent groups of two hexadecimal figures.

⚠ To define the colors in hexadecimal, the figures must be preceded by a dollar sign ($).

An example of the section [DefPalette] from the file nslib.ini. Representation of the colors in decimal figures:

```
[DefPalette]
Color1=57206238
Color2=0,0,0
...
Color18=109199125
```

Section [SheetTreeBox]

Sets the default colors of the lines of SheetTreeBox and Ldata (NatStar only) when the corresponding color is left at `Default` during creation of the custom control. The colors are the integer numbers of the constants COL_*%.

```
BackCol
```

Background color of the lines that are not selected and are not a title.

```
ForeCol
```

Text color of the lines that are not selected and are not a title.

```
TitleBackCol
```

Background color of the title lines.

```
TitleForeCol
```

Text color of the title lines.

```
SelBackCol
```

Background color of the selected lines.

```
SelForeCol
```

Text color of the selected lines.

The instruction SHEET_SETDEFCOLORS which takes these 6 colors (in the same order) allows default values to be given, different from those defined in NSLIB.INI for a given application. If the value is defined as -1 for a color, the value is taken from NSLIB.INI, if the value is given as -2, it is the default value that will have been used in the absence of any definition taken from NSLIB.INI (respectively COL_WINDOW%, COL_WINDOWTEXT%, COL_BUTTONMIDDLE%, COL_BUTTONTEXT%, COL_HILITEBACKGROUND% and COL_HILITEFOREGROUND%).

Default values

```
[SheetTreeBox]
BackCol = 41
ForeCol = 44
TitleBackCol = 21
TitleForeCol = 54
SelBackCol = 29
SelForeCol = 28
```

Customization of the user interface

XWindow targets

The behavior of the XWindow libraries can be customized in order to obtain a user interface corresponding, for example, to what the end users are accustomed to.

Keyboard

The correspondence between key sequences and the actions carried out may be completely redefined. This mechanism can be useful when some keys used by portable libraries do not exist on your keyboard, or even when the default correspondence is not suitable.

A key redefinition section of the form `[ControlType.Translations]`is associated with each type of control or window. An action is redefined using a form element:

```
action=[modificator-key+ [modificator-key+]
        ...]key
```

For example, in order to associate [Ctrl+F5] to the action « Opening/Closing of the list» for all the Combo Boxes in your application, you only have to add the following lines to the NSLIB.INI file:

```
[Combo.Translations]
ToggleChild=Ctrl+F5
```

A detailed list of key redefinitions may be found in Appendix B of this manual.

System colors

X Window portable libraries use 16 basic colors to draw controls or windows. Each control may have a color defined by the development environment or use the default color for this type of control.

By default, each element is drawn with the system color contained in a range going from COL_BACKGROUND% to COL_HELPHILITE% (color constant of the NCL language).

Each system color is defined as one of 16 basic colors (COL_BLACK% ... constant). COL_WHITE% of the NCL language). It is possible to change these definitions using the NSLIB.INI file.

To redefine a system color, all that has to be done is to add the following section to NSLIB.INI:

```
[ColorPalette]
system-color=base-color
```

For example, to redefine the color of the title bar of the active child windows to black on a blue background:

```
[ColorPalette]
ActiveTitleText=Black
ActiveTitle=Blue
```

An exhaustive list of the basic and system colors can be found in Appendix B of the « Porting applications » manual.

Labels

A certain number of controls are created directly by X Window portable libraries with labels in English. It relates to *Push-buttons*, message boxes and child window menu system items.

It is possible to customize the text of these items using the NSLIB.INI file. This mechanism allows a user interface to be entirely in French for example.

Menu system item labels are redefined in the section `[Menu.System.Text]`. Dialog box button labels are redefined in the section `[Button.Text]`.

Menu system in French

```
[Menu.System.Text]
Restore=~Restauration
Move=~Déplacement
Size=D~imensionnement
Minimize=Réd~uction
Maximize=~Agrandissement
Close=Arrêt/~Fermeture
```

The ~ character (tilde) allows a keyboard shortcut to be defined.

A full list of redefinable labels can be found in Appendix A of this manual.

Fonts

The `[Fonts]` section allows fonts to be mapped and the default font to be defined.

To indicate the default font in nslib.ini

Use the item `Default` in the section `[Fonts]`.

Example

```
[Fonts]
Default=-adobe-times-medium-r-*--*-120-100-100-*-*-iso8859-1
```

Automatic choice algorithm

If the name of the default font is not defined in the NSLIB.INI file, X Window libraries try to load the following fonts (in order):

1.

```
-*-menu-medium-r-normal-*-*-120-*-*-p-*-iso8859-1
```

2.

```
-adobe-helvetica-bold-r-normal--*-120-*-*-p-*-
iso8859-1
```

3.

```
fixed
```

4.

```
-*-*-*-r-*-*-*-120-*-*-*-*-iso8859-1
```

Printing system

Before using the printing system, the configuration of the printers connected to your system must be described. For each printer, its type and the print command to be used should be defined.

The section `[Printer]` allows printers connected to the system to be described.

This description is contained in the NSLIB.INI FILE file.

- UNIX system:

```
; definition of the print’s queue
;
[Printer]
HPLaser1=QUEUE=‘lp0’
HPLaser1=TYPE=‘postscript’
HPLaser1=RAW=‘lp -d %%QUEUE %%FILE’
HPLaser1=TEXT=‘pr %%FILE || lp -d %%QUEUE’

Default=HPLaser1
```

- VMS system:

```
[Printer]
HPLaser1=QUEUE=‘SYS$PRINT’
HPLaser1=TYPE=‘postscript’
HPLaser1=RAW=‘PRINT/PASSALL/DELETE/NOIDENT/QUEUE=%%QUEUE %%FILE.’
HPLaser1=TEXT=‘PRINT/DELETE/NOIDENT/QUEUE=%%QUEUE %%FILE.’’

Default=HPLaser1
```

Syntax

`[Printer]` is the name of the section describing the printers.

`HPLaser1` will be the name of the printer that appears with the command `REP_INFO$ (prn%, IDX_NAME%)`, this name can include any character string.

Printer description commands can be on the same line, separated by the « , » character or on different lines. Each command should start with the name of the printer followed by « = ».

- The UNIX queue to be used by the printer

```
HPLaser1=QUEUE='lp0'
```

- Printer model

```
HPLaser1=TYPE='postscript'
```

- Command for printing text files (with formatting)

```
HPLaser1=TEXT='pr %%FILE || lp -d %%QUEUE'
```

- Command for printing binary files (without formatting)

```
HPLaser1=RAW='lp -d %%QUEUE %%FILE'
```

- The default printer that will appear with the command REP_DEFAULT% (). This name **must** be one of the names declared previously.

```
Default=HPLaser1
```

☞ In the two commands TEXT and RAW:

- `%%FILE` designates the file to be printed.
- `%%QUEUE` designates the queue defined previously.

Miscellaneous

The section `[System]` allows various options to do with the behavior of portable libraries to be defined.

- The option `Look` allows the definition of the type of user interface to be emulated: `PM` (default interface) or `MOTIF`.

  To emulate the Motif user interface, add the following lines to your NSLIB.INI file:

```
[System]
Look=MOTIF
```

- The option `ForceTitleBar` allows the systematic appearance of a title bar on all the user interface windows even if, earlier, the option *Title bar* option was un-checked.

  Windows without title bars can cause problems with some X Window emulators under Windows (X Vision notably).

  To force a title bar in all windows, add the following line to the section `[System]`:

```
[System]
ForceTitleBar=True
```

- The options `XRatio` and `YRatio` allow definition of a percentage that allows all the windows of the application to be re-sized.

  Example

```
Xratio=150
YRatio=150
```

Target characters

For target characters, the NSLIB.INI file additionally allows the configuration of certain elements of the targets, such as the actions that can be produced via the keyboard or the default colors, and the localization of the system parts of the applications.

List of sections

Sections	**Redefined elements**
`[System.Translations]`	System kernel
`[Frame.Translations]`	System kernel
`[Menu.Translations]`	System menus
`[Dialog.Translations]`	Dialogue boxes
`[Entry.Translations]`	Entry Fields
`[Push.Translations]`	Push buttons
`[List.Translations]`	List boxes
`[Check.Translations]`	Check boxes
`[Radio.Translations]`	Radio buttons
`[Combo.Translations]`	Combo boxes
`[ComboEntry.Translations]`	Combo boxes with Entry Field
`[MLE.Translations]`	Multi line Entry Fields
`[Scroll.Translations]`	Scroll bars
`[ColorPalette]`	System colors
`[Spooler]`	Print queue
`[Button.text]`	System parts texts: Push-buttons,

	messages boxes
`[Menu.system.text]`	Menu system texts
`[Tasklist.text]`	Task list texts
`[Keypad]`	Additional functions (VMS and Unix)
`[Entry]`	Additional functions

Keyboard configuration

For each target, you can redefine the actions triggered from the keyboard.

A key redefinition section `[`*`WinCtrl`*`.Translations]`is associated with each type of control or window. The following line redefines an action:

```
action=[mod. key+[+mod. key]...]touche
```

For example, under UNIX or VMS, the following lines allow [Gold]+[F4] to be associated with the action « opening the list » :

```
[Combo.Translations]
ToggleChild=Gold+F4
```

Actions that may be triggered using the keyboard are described in the appendices « Use of the keyboard » in the « Porting of applications » manual.

The list of sections relating to key redefinitions, shared by all targets, may be found in Appendix B of this manual.

List of keyboard keys

Keys

Modifier keys	**Description**
Ctrl	[Ctrl] key depressed[(1)]
Alt	[Alt] key depressed
Gold	[PF1] key depressed.
	This key may be redefined.
Left	Left
Right	Right
Up	Up
Down	Down
PageUp	Page up
PageDown	Page down
Space	Space bar
Tab	Tabulation
NewLine	[Enter] key on the numeric keypad
Insert	[Insert] key
Delete	[Delete] key
Backspace	⌫ key
Enter	Return key
F7 -->F20	On a VT220 keyboard
F1 --> F12	On a keyboard with a 3270 emulator
PF1 --> PF4	Above the numeric keypad
KP0 --> KP9	Keys 0 to 9 of the numeric keypad if it is in application mode.
Minus	Minus key on the numeric keypad in application mode

Keys	Description
Comma	Comma key on the numeric keypad in application mode
Period	Full stop/period key on the numeric keypad in application mode
Select	Central mini keypad
Find	Central mini keypad
Help	Help key or [F15]
Execute	Do Key or [F16]

NOTE : **(1)** Only works with letters of the alphabet

Color configuration

The section `[ColorPalette]` redefines the default colors of some parts of the windows and controls.

Under UNIX or VMS, each system color has a name with a corresponding color + an attribute.

To redefine a system color, all that has to be done is to add the following line in the section `[ColorPalette]`:

```
systemcolors =color [+ attribute[+attribute]...]
```

A list of system colors can be found in appendix B of the « Porting of applications » manual.

Example

```
Tilde =Black+Underline
```

Most of the elements are drawn using two colors: a foreground color and a background color. The attributes of these two colors are cumulative.

For example, if the background is defined with White+Underline and the foreground with Black+DoubleBright, the letters will be underlined and double bright.

Only the colors Black and White are available.

List of Attributes

Attribute	Description
Underline	Character underlined
Inverse	Inverse video
DoubleBright	Double brilliance
Blink	Flashing

☞ If at Runtime, the NSLIB.INI file is not the same as the one used during Design, the colors defined may be different.

Labels

The text of the system parts, in English by default, can be redefined in the sections `[Button.text]`, `[Menu.system.text]` and `[Tasklist.text]`.

To redefine a text, all that has to be done is to add the following line in the appropriate section:

```
SystemText=Text
```

An example of the localization of system parts texts in French is provided in Appendix A of this manual.

An example of translation allowing localization of the text of the system menus in French.

```
[Menu.system.text]
Restore=~Restauration
Move=~Déplacement
Size=D~imensionnement
Minimize=Réd~uction
Maximize=~Agrandissement
Close=Arrêt/~Fermeture
TaskManager=~Liste des fenêtres
```

☞ The ~ character (tilde) allows a keyboard shortcut to be defined.

Additional functions

Additional functions are available for each target, allowing changes to be made to keyboard behavior as well as the display of *Entry Fields*.

Additional functions

Functions	**Values**	**Description**
Section [Entry]		
Background	Any character	Defines the background character of entry fields
InsertMode	TRUE	Defines the keyboard state at the start of an application
FocusSelect	TRUE	Specifies if the text of an *entry field* should be selected when it takes the focus
Section [Keypad]		
Numeric	"Normal" or "Application"	Allows the keys on the numeric keypad to be used as function keys
Goldkey	Any function key	defines the [Gold] key.
Type	"VT220" or "Emul3270"	Use of function keys in VT mode or 3270 emulation

Chapter 2

NSDKCFG.INI and NSDKLOC.INI Files

This chapter presents the NSDKCFG.INI and NSDKLOC.INI configuration files which replace the former NS-DK.INI file.

You will find in this chapter

- The different sections of the NSDKCFG.INI file.
- The sections and parameters.
- The syntax of the NSDKCFG.INI file.

Contents

Introduction

The NS-DK.INI file has been replaced with two configuration files: NSDKCFG.INI (corresponds to your project's global configuration file) and NSDKLOC.INI (corresponds to a project's local configuration file).

The file NSDKCFG.INI is provided with NS-DK. The purpose of this file is to be shared on a networked NS-DK installation, for example. The NSDKCFG.INI file is usually modified by an NS-DK installation administrator or a project manager.

The file NSDKLOC.INI is not provided with NS-DK. It's a configuration file specific to each user, who creates it and modifies it.

The purpose of the NSDKCFG.INI and NSDKLOC.INI files is to specify a range of tools and parameters that are displayed as defaults in the various generation boxes. These tools are grouped into toolsets, whose names are displayed by the *Toolset* field in the *Create new Configuration* and *Configurations* boxes. The corresponding details for the toolset selected from this box are displayed in the other generation boxes:

- The start-up parameters for NS-Gen are displayed as defaults in the *Configurations* box at the *Generator* tab.
- The compiler and its parameters are displayed as defaults in the *Configurations* box at the *Compiler/Linker* tab.
- The linker, ImpLib utility, resource compiler and their parameters are displayed as defaults in the *Configurations* box at the *Compiler/Linker* tab

The NSDKCFG.INI file allows you to generate your applications rapidly and easily by selecting only one toolset. The corresponding tools and their start-up parameters will automatically appear in the generation boxes for each new application.

The NSDKCFG.INI file cannot be updated from the various generation boxes. Any changes made via these boxes are only saved in the current .N_C configuration file and apply specifically to the generation context for the current application.

To update the NSDKCFG.INI and NSDKLOC.INI files, you will need to use a text editor and comply with the syntax shown below. In this way, you can define a new toolset containing the generation tools and start-up parameters that you normally use.

NS-INI environment variable

Nat System advises you to set the NS-INI environment variable to point to a local user directory, with an NS-DK installation that may be networked.

So, the NSDKCFG.INI file is automatically found in the NS-DK installation, while the NSDKLOC.INI file is created in the local directory pointed to by NS-INI.

The NSDKCFG.INI file is looked for first in the following directories:

1. In the directory pointed to by the NS-INI environment variable (without the name of the project, therefore).

2. If it doesn't exist in the directory pointed to by the NS-INI environment variable it is looked for in the installation */ini* directory.

3. If it doesn't exist in the /ini directory, it is created in the directory pointed to by NS-INI.

4. If the directory pointed to by the NS-INI environment variable doesn't exist or if NS-INI is not defined, it is created in the installation */ini* directory.

The file NSDKLOC.INI is first looked for in the directory pointed to by NS-INI:

1. If it doesn't exist in the directory pointed to by the NS-INI environment variable it is created in the directory pointed to by NS-INI.

2. If the directory pointed to by NS-INI doesn't exist or if NS-INI is not defined, it is looked for in the installation INI directory.

3. If it doesn't exist in the installation INI directory, it is created in the installation INI directory.

If the NS-DK installation is shared on a network, and if the variable NS-INI has not been defined for several users, there may be a risk of collision in the NSDKLOC.INI file that becomes a shared file (there is no protection against simultaneous writing by several users).

Conversely, if the NS-DK installation is distributed on each station, and if the NS-INI variable has not been defined, each user will create an NSDKLOC.INI file in the INI directory of the local NS-DK installation.

Syntax for the NSDKCFG.INI file

The NSDKCFG.INI file is made up of sections. Each section contains a list of parameters that define a toolset. These toolset parameters specify the names, pathnames and default parameters of the most commonly used compilers, linkers, ImpLib utilities and resource compilers.

This file complies with the following syntax:

```
 [Toolsets]
<ToolsetName1>=
<ToolsetName2>=
...

[<ToolsetName1>]
Generator_Options = <OptionsList>
Compiler = <PathName>
Compiler_Options = <OptionsList>
Linker = <PathName>
EXE_Linker_Options = <OptionsList>
DLL_Linker_Options = <OptionsList>
EXE_Libraries = <LibrariesList>
DLL_Libraries = <LibrariesList>
Implib = <PathName>
Implib_Options = <OptionsList>
Resource_Compiler = <PathName>
Resource_Compiler_Options = <OptionsList>

[<ToolsetName2>]
Generator_Options = <OptionsList>
Compiler = <PathName>
Compiler_Options = <OptionsList>
Linker = <PathName>
EXE_Linker_Options = <OptionsList>
DLL_Linker_Options = <OptionsList>
EXE_Libraries = <LibrariesList>
DLL_Libraries = <LibrariesList>
Implib = <PathName>
Implib_Options = <OptionsList>
Resource_Compiler = <PathName>
Resource_Compiler_Options = <OptionsList>

...
```

[Toolsets] Section

This section lists the toolsets that will be displayed by the Toolset field in the *Create new configuration* box. The syntax required is:

```
<ToolsetName>=
```

where

<ToolsetName>	stands for the logical toolset name. This name is determined by the person who updates the NSDKCFG.INI file.

This name must appear later as the heading of another section which specifies the names and parameters of the tools used to build an application.

The toolset name is displayed in the Toolset list of the *Project generation* box.

Example

```
MSVC-Win32=
```

contains all the tools required to build an application that runs under the Windows 32-bit environment, based on the MSVC utilities (Microsoft Visual C++).

[<ToolsetName>] Sections

These sections list the following information for each toolset defined in the [Toolsets] section:

- The default generator options displayed in the *Configurations* box at the *Generator* tab.
- The default compiler name and parameters displayed in the *Configurations* box at the *Compiler/Linker* tab.
- The default tool names and associated parameters displayed in the *Configurations* box at the *Compiler/Linker* tab.

This type of section has the default following syntax:

```
[<ToolsetName>]
Generator_Options = <OptionsList>
Compiler = <PathName>
Compiler_Options = <OptionsList>
Linker = <PathName>
EXE_Linker_Options = <OptionsList>
DLL_Linker_Options = <OptionsList>
EXE_Libraries = <LibrariesList>
DLL_Libraries = <LibrariesList>
Implib = <PathName>
Implib_Options = <OptionsList>
Resource_Compiler = <PathName>
Resource_Compiler_Options = <OptionsList>
```

But you can insert new components corresponding to the paths directories used to store all the generated files:

```
[<ToolsetName>]
…
Sources_Directory=<PathDirectory>
Includes_Directory=<PathDirectory>
Objects_Directory= <PathDirectory>
Binaries_Directory= <PathDirectory>
DLLs_Directory= <PathDirectory>
Libraries_Directory= <PathDirectory>
```

The following is a description of each line:

Generator_Options

Start-up options for the NS-Gen generator.

These are the default options displayed by the *Generator line command options* field in the *Configurations* box at the *Generator* tab.

Compiler

Full pathname of the compiler.

This is the default pathname displayed by the *Path* field in the *Configurations* box at the *Compiler/Linker* tab.

Compiler_Options

Start-up options for the compiler.

These options depend on the compiler used. They are the default options displayed by the *Options* field in *Compiler* group of the *Configurations* box at the *Compiler/Linker* tab.

Linker

Full pathname of the linker.

This is the default pathname displayed by the *Path* field in *Linker* group of the *Configurations* box at the *Compiler/Linker* tab.

EXE_Linker_Options

Start-up options for the linker when generating an executable (*Standalone executable* option checked in the *Configurations* box at the *Generator* tab).

These options depend on the linker used. They are the default options displayed by the *EXE Options* field in the *Configurations* box at the *Compiler/Linker* tab.

DLL_Linker_Options

Start-up options for the linker when generating DLLs (*Single DLL* or *Multiple DLLs* options checked in the *Configurations* box at the *Generator* tab).

These options depend on the linker used. They are the default options displayed by the DLL *Options* field in the *DLL linker* of the *Configurations* box at the *Compiler/Linker* tab.

EXE_Libraries

Libraries used by the linker when generating an executable (*Standalone executable* option checked in the *Configurations* box at the *Generator* tab).

These libraries depend on the linker used. They are the default libraries displayed by the *EXE Libraries* field in the *Configurations* box at the *Compiler/Linker* tab.

DLL_Libraries

Libraries used by the linker when generating DLLs (*Generate DLLs* option checked in the *Setup generator* box).

These libraries depend on the linker used. They are the default libraries displayed by the *DLL Libraries* field in the *Configurations* box at the *Compiler/Linker* tab.

Implib

Full pathname of the ImpLib utility.

When you generate DLLs, ImpLib is used to generate the LIB file that contains the names of the procedures in the associated DLL. The LIB file is used when you link an external program that uses the DLL.

This is the default pathname displayed by the Implib Path field in the *Configurations* box at the *Compiler/Linker* tab.

Implib_Options

Start-up options for the ImpLib utility.

These options depend on the Implib utility used. They are the default options displayed by the *Implib Options* field in the *Configurations* box at the *Compiler/Linker* tab.

Resource_Compiler

Full pathname of the resource compiler, which is used to associate an icon with the generated executable.

This is the default pathname displayed by the *Resource Compiler Path* field in the *Configurations* box at the *Compiler/Linker* tab.

Resource_Compiler_Options

Start-up options for the resource compiler.

These options depend on the resource compiler used. They are the default options displayed by the *ResOpts* field in the *Configurations* box at the *Compiler/Linker* tab.

Sources_Directory

Path directory used to store the source files (.C) generated by NS-Gen.

This path is displayed by default in the *Sources* field of the *Directories* tab of the *Configurations* box.

Includes_Directory

Path directory used to store the include files (.H and .D) generated by NS-Gen.

This path is displayed by default in the *Includes* field of the *Directories* tab of the *Configurations* box.

Objects_Directory

Path directory used to store the object files (.OBJ) generated by the compiler. It also stores two files generated by NS-Gen: the definition file (.DEF) and the .NSE file that contains the names of any exported functions.

This path is displayed by default in the *Objects* field of the *Directories* tab of the *Configurations* box.

Binaries_Directory

Path directory used to store the executable file (.EXE) generated by the linker as well the external resource files (.RES) generated by NS-Gen.

This path is displayed by default in the *Binaries* field of the *Directories* tab of the *Configurations* box.

DLLs_Directory

Path directory used to store the DLLs generated by the linker.

This path is displayed by default in the *DLLs* field of the *Directories* tab of the *Configurations* box.

Libraries_Directory

Directory used to store the LIB files generated by the ImpLib utility.

This path is displayed by default in the *Libraries* field of the *Directories* tab of the *Configurations* box.

Example

```
[MSVC-Win32]
Generator_Options = /TOOLKIND:MSVC32
Compiler = CL.EXE
Compiler_Options = /c /W3 /Gs /Zp /DWIN32
Linker = LINK.EXE
EXE_Linker_Options = /SUBSYSTEM:WINDOWS
DLL_Linker_Options = /SUBSYSTEM:WINDOWS /DLL
EXE_Libraries = NSLIB.LIB
DLL_Libraries = NSLIB.LIB
Implib = LIB.EXE
Implib_Options = /MACHINE:IX86
Resource_Compiler =
Resource_Compiler_Options =
Sources_Directory=C:\NSDK\exemple\C
Includes_Directory= C:\NSDK\exemple\H
Objects_Directory= C:\NSDK\exemple\OBJ
Binaries_Directory= C:\NSDK\exemple\BIN
DLLs_Directory= C:\NSDK\exemple\DLL
Libraries_Directory= C:\NSDK\exemple\LIB
```

Notes

The syntax for each parameter is the same as the syntax for the corresponding field in the Generator and Compiler/Linker tabs of the Configurations box.

If a parameter is missing or not followed by a value, the field in the corresponding dialog box will be blank.

If NS-Design is in use, any changes made to the NSDKCFG.INI file will not take effect until it has been stopped.

If the NSDKCFG.INI file contains a syntax error, a message will be displayed when NS-DK is started up and the Toolset field in the Create new configuration box will be disabled.

Appendix A Labels

You will find in this chapter

Various tables presenting the label redefinition sections associated with graphical objects such as menus or *Push buttons*, that are configured in the NSLIB.INI file.

☞ All the sections presented in the following apply equally to X Window targets and character targets.

Contents

Labels

The text of the system parts, in English by default, can be redefined in the sections `[Button.text]`, `[Menu.system.text]` and `[Tasklist.text]`.

To redefine a text, all that has to be done is to add the following line to the appropriate section:

```
Systemtext =Text
```

An example of the localization of system parts texts in French is provided in Appendix B of this manual.

An example of translation allowing localization of the text of the system menus in French.

```
[Menu.system.text]
Restore=~Restauration
Move=~Déplacement
Size=D~imensionnement
Minimize=Réd~uction
Maximize=~Agrandissement
Close=Arrêt/~Fermeture
TaskManager=~Liste des fenêtres
```

The ~ character (tilde) allows a keyboard shortcut to be defined.

Redefinition of labels in the NSLIB.INI file

Under Windows 32, redefinition is automatic. The text depends on the language of the operating system. The following paragraphs are not relevant, therefore, to Windows 32.

Section [Button.text]

This section allows changes to be made to the text of *Push buttons* contained in message boxes created by the MESSAGE, ASK2%, ASK3% and MESSAGE% (NSMisc library) functions.

System text	Text in French	Description
Ok	Valider	*Push button Ok*
Enter	Entrée	*Push button Enter*
Yes	~Oui	*Push button Yes*
No	~Non	*Push button No*
Abort	~Abandon	*Push button Abort*
Retry	~Réessayer	*Push button Retry*
Ignore	~Ignorer	*Push button Ignore*
Cancel	Annuler	*Push button Cancel*
Help	Aide	*Push button Help*

Section [Menu.System.Text]

This section allows changes to be made to the text of items in the menu system created by portable libraries for daughter windows (created by `OPEN nom, handle- parent`).

System text	Text in French	Description

Move	~Déplacement	*Move* menu item
Size	D~imensionnement	*Size* menu item
Minimize	Réd~uction	*Minimize* menu item
Restore	~Restauration	*Restore* menu item
Maximize	~Agrandissement	*Maximize* menu item
Close	Arrêt/~Fermeture	*Close* menu item
TaskManager	~Liste des fenêtres	*Window list* menu item (character targets)

Section [Tasklist.text]

This section allows changes to be made to the text of items in the task list.

☞ Only for character targets.

System text	Text in French	Description
Title	Liste des tâches	List title
Activate	~Activer	*Push button Activate*
Close	~Fermer	*Push button Close*

Appendix B Redefinition of keys

You will find in this chapter

Various tables presenting the sections handling redefinition of the actions associated with keys on the keyboard that are configured in the NSLIB.INI file.

☞ All the sections presented in the following apply equally to X Window targets and Character targets. However, the third column in each of the tables, "Default X Window sequence", is only relevant to X Window targets.

Contents

Redefinition of keys in the NSLIB.INI file

Section [Frame.Translations]

This section allows redefinition of actions common to all types of windows (*Window, Dialog, List Box, Edit* ...)

Description of the action	Name of the action	Default X Window sequence
Access to the menu	EnterMenu	F10
Access to the system menu	EnterSysMenu	Alt + Espace
Restore the window (1)	(2)	Alt + F5
Move the window (1)	(2)	Alt + F7
Resize the window (1)	(2)	Alt + F8
Minimize the window (1)	(2)	Alt + F9
Maximize the window (1)	(2)	Alt + F10
Close the window (1)	(2)	Alt + F4

NOTES:

(1) These actions are only handled for child windows (opened by `OPEN nom, handle-parent`). The Motif Window Manager (mwm) handles the menu system for main windows.

(2) These actions cannot be redefined.

☞ In the case of MDI child windows, the Alt modifier is replaced by Ctrl.

Section [Menus.Translation]

This section allows redefinition of the actions of menu and sub-menu bars.

Description of the action	**Name of the action**	**Default X Window sequence**
Close menu	ExitMenu	F10
Close a level	CloseMenu	Escape
Execute a choice	Execute	Enter or Newline
Previous sub-menu or previous item via a sub-menu	Up	Up
Next sub-menu or open a sub-menu from the main menu	Down	
Go to the next sub-menu or open a sub-menu from a sub-menu	Right	
Go to the previous sub-menu or open a sub-menu from a sub-menu	Left	
First menu or item	Home	Home
Last menu or item	End	End
Help	**(1)**	F1

NOTE:

(1) This action cannot be redefined.

Section [Dialog.Translations]

This section allows redefinition of the actions particular to dialog boxes (*Dialog* type windows).

Description of the action	**Name of the action**	**Default X Window sequence**
Next control	NextControl	Tab
Previous control	PreviousControl	Shift + Tab
Default push button	OKButton	Enter or Newline
Exit push button	EscapeButton	Escape
Help	Help	F1

Section [Entry.Translations]

This section allows redefinition of the actions of controls of the *entry field* type.

Description of the action	**Name of the action**	**Default X Window sequence**
Previous character	PreviousChar	
Next character	NextChar	
Previous word	PreviousWord	Ctrl +
Next word	NextWord	Ctrl +
Start	Home	Home
End	End	End
Delete current character	DeleteCurrent	Delete
Delete previous character	DeletePrevious	Backspace

Cut	Cut	Delete
Copy	Copy	Ctrl + Insert
Paste	Paste	Shift + Insert
Insert/Overtype	ToggleInsert	Insert
Spin button / down	Spin down	
Spin button / up	Spin up	
Help	Help	F1

Section [Push.Translations]

This section allows redefinition of the actions of controls of the *push button* type.

Description of the action	**Name of the action**	**Default X Window sequence**
Press the push button	Press	Space
Execute the push button	Execute	Enter or Newline

Section [List.Translations]

This section allows redefinition of the actions of controls of the *list box* type.

Description of the action	**Name of the action**	**Default X Window sequence**
Select/Deselect	Toggle	Space
Next page	NextPage	Page down
Previous page	PreviousPage	Page up
First line	Home	Home
Last line	End	End

Previous line	PreviousLine	
Next line	NextLine	
Previous line	PreviousLine	Shift +
Next line	NextLine	Shift +
Scroll right	ScrollRight	
Scroll left	ScrollLeft	
Go to right of list	ScrollEnd	Ctrl + End
Go to left of list	ScrollHome	Ctrl + Home
Execute a line	Execute	Enter or Newline

☞ It is also possible to go to a line by typing the first letter of the line. If several lines start with the same letter, you go through them from top to bottom.

Sections [Combo.Translations] and [ComboEntry.Translations]

These sections allow redefinition of the actions of controls of the *combo box* and *combo box with entry field* type. They are added to the actions defined for controls of the *list box* and *entry field* type (for controls of the *combo box with entry field* type).

Description of the action	**Name of the action**	**Default X Window sequence**
Open / Close list	ToggleChild	Alt +
Open / Close list	ToggleChild	F4
Close list	CloseChild	Escape

Section [Radio.Translations]

This section allows redefinition of the actions of controls of the *radio button* type.

Description of the action	**Name of the action**	**Default X Window sequence**
Next radio button	Next	
Previous radio button	Previous	
Select	Select	Space

Section [Check.Translations]

This section allows redefinition of the actions of controls of the *check box* type.

Description of the action	**Name of the action**	**Default X Window sequence**
Change state	Toggle	Space

Section [MLE.Translations]

This section allows redefinition of the actions of controls of the *Multi Line Entry field* type.

Description of the action	**Name of the action**	**Default X Window sequence**
Next character	NextChar	
Previous character	PreviousChar	
Next word	NextWord	Ctrl +
Previous word	PreviousWord	Ctrl +
Next line	NextLine	
Previous line	Previous line	

Next page down	NextVertPage	Page down
Next page up	PrevVertPage	Page up
Next page across	NextHrzPage	Ctrl + Page down
Previous page across	PrevHrzPage	Ctrl + Page up
Start of line	BeginLine	Home
End of line	EndLine	End
Start of list	BeginDoc	Ctrl + Home
End of list	EndDoc	Ctrl + End
Cut	Cut	Shift + Delete
Copy	Copy	Ctrl + Insert
Paste	Paste	Shift + Insert
Insert/Overtype	ToggleInsert	Insert
New line	NewLine	Newline or Enter
Delete current character	DeleteCurrent	Delete
Delete previous character	DeletePrevious	Backspace
Undo	Undo	Alt + Backspace

Section [Scroll.Translations]

This section allows redefinition of the actions of controls of the *scroll bar* type.

Description of the action	Name of the action	Default X Window sequence
Previous line	LineUp	
Next line	LineDown	
Previous page	PageDown	Page down
Next page	PageUp	Page up
Start	Home	Home
Start	Home	---
End	End	End
End	End	---