

NS-DK

Version 5.00 Edition 2

New features of NS-DK 5.00

Information in this document is subject to change without notice as a result of changes in the product. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is illegal to reproduce the software on any medium unless specifically authorized within the terms of the agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose other than the purchaser's personal use without the written permission of Nat System.

© 2007 Nat System. All rights reserved

The Nat System name and logo are registered trademarks of Nat System.

All other trademarks mentioned in this document are registered trademarks of their respective companies.

Contents

Organization of the manuel.....	v
Conventions	vi

Chapter 1 General Presentation

Technological environment.....	1-4
Windows.....	1-4
Unix	1-4
Linux	1-5
Database	1-5
New NS-Design interface	1-6
New organization of the workspace	1-6
Resource browser	1-7
The property pane.....	1-9
NCL editor.....	1-11
Macros.....	1-13
New organization of menus and tool bars	1-14
New keyboard shortcuts	1-15
Modification of some dialog boxes	1-18
Deleting historical limitations	1-21
XNP file.....	1-21
Size of the resource name and new resources	1-21
64 KB limitations	1-21
Improvement of the NCL language.....	1-22
New type : DynStr.....	1-22
Extension of the POS% function.....	1-22
Trace API	1-22
Changes to NCL services	1-23
Passing parameters to an event.....	1-24
Enhancement of NSWIN.....	1-24
Graphical controls and display.....	1-26
Anchoring controls	1-26
NS-ScreenExploser	1-30
SheetBox control	1-30

External menu control	1-30
The NSDKCFG.INI and NSDKLOC.INI files	1-32
Generation and debugging	1-34
Limitation	1-34
Automatic trace generator	1-34
NS-DebugMem	1-34
Optimization for 64-bits	1-34
DLL names	1-35

Chapter 2 New fonctions and instructions

NSWIN library	2-3
Reading/Writing in the Windows Registry	2-3
Managing the mouse pointer position	2-3
Attachement of windows	2-3
NSMisc library	2-19
Managing traces	2-19
Managing the anchoring controls	2-19
NSSEQ library	2-28

Chapter 3 NCL language

Verbs	3-3
-------------	-----

Organization of the manuel

NS-DK 5.00 contains some new features, and several new or modified APIs.

This document fully describes these enhancements and refers you to the appropriate manuals in the NS-DK document set for further information.

This manual is divided into three chapters.

Chapter 1	General presentation
------------------	-----------------------------

This chapter introduces the new features of NS-DK 5.00.

Chapter 2	New functions and instructions
------------------	---------------------------------------

This chapter introduces the new functions and instructions of NS-DK 5.00.

Chapter 3	NCL language
------------------	---------------------

This chapter introduces the new commands NCL language.

Conventions

Typographic conventions

Important term	Important terms are printed in bold .
<i>Interface component</i>	The names of windows, dialog boxes, controls, buttons, menus and options are printed in <i>italics</i> .
[F9]	Function 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

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

General Presentation



NS-DK 5.00 integrates new features to optimize the applications developed with this tool. Thus, Nat System fulfills the technological and technical requirements of its users.

You will find in this chapter

- The new interface of NS-DK
- The technological environments supported by NS-DK 5.00.
- The new functionalities of some controls.
- The new features in the NCL language.
- The anchoring of controls.

Table of contents

Technological environment.....	1-4
Windows	1-4
Unix	1-4
Linux	1-5
Database	1-5
New NS-Design interface	1-6
New organization of the workspace	1-6
Resource browser	1-7
The property pane	1-9
NCL editor	1-11
Macros	1-13
New organization of menus and tool bars	1-14
The File menu	1-14
The Edit menu	1-14
The View menu	1-14
The Build menu	1-15
The Options menu	1-15
The Controls menu	1-15
New keyboard shortcuts	1-15
Keyboard shortcuts of the File menu	1-15
Keyboard shortcuts of the Edit menu	1-15
Keyboard shortcuts of the View menu	1-16
Keyboard shortcuts of the View menu	1-16
Actions of copy/paste	1-17
Other manipulations in the code editor	1-17
Modification of some dialog boxes	1-18
Deleting historical limitations	1-21
XNP file	1-21
Size of the resource name and new resources	1-21
64 KB limitations	1-21
Improvement of the NCL language	1-22
New type : DynStr	1-22
Extension of the POS% function	1-22
Trace API	1-22
Changes to NCL services	1-23
Passing parameters to an event	1-24

Enhancement of NSWIN	1-24
Management of the Windows Registry	1-25
Management of the mouse pointer position	1-25
Graphical controls and display	1-26
Anchoring controls	1-26
Principle	1-26
Examples of use	1-28
Use	1-28
Development recommendations	1-29
Dynamic changes	1-29
See also	1-29
Comment	1-29
NS-ScreenExploser	1-30
SheetBox control	1-30
External menu control	1-30
The NSDKCFG.INI and NSDKLOC.INI files	1-32
Generation and debugging	1-34
Limitation	1-34
Automatic trace generator	1-34
NS-DebugMem	1-34
Optimization for 64-bits	1-34
DLL names	1-35

Technological environment

Our priority target for the client part remains the 32-bit Windows environment. However, in order to meet the expectations of our clientele, in particular in the field of public agencies, a new Linux target was added to this new version. A NS-DK application will be able to run, therefore, on a Linux system, with just a few portability constraints.

As far as Unix servers are concerned, for customers who also use NS-DK as a tool for developing their batches, or in client-server mode, 64-bit platforms will be supported.

In addition, support for Linux and MySQL enables us to offer an alternative to Windows for “small” servers.

As far as databases are concerned, support for new databases will be integrated, in particular for MySQL, Oracle 10G and SQL Server 2005.

The new Oracle 10G access driver introduces CLOB (Character Large OBject) types which for Oracle should replace LongRaw types as well as BLOB (Binary Large OBject) types.

Windows

Support for Windows platforms will initially be limited to 32-bit. The following versions will be supported:

- Windows XP SP1 and SP2,
- Windows 2003,
- Windows 2000,
- Windows NT 4.0,
- Windows 98 SE.

Unix

Unix target servers will now be available in 32 and 64- bit. The following servers will be available:

- Aix 5.2 in 32 and 5.3 in 64 bits,
- Hp-UX 11.1 PA-Risc 2 in 32 and 64 bits,
- Sun Solaris 10 in 32 and 64 bits.

Linux

NS-DK 5.00 will have two Linux Red Hat Enterprise 3.0 targets for Intel processors, a 32-bit target server permitting the creation of batches, as well as client-server applications, and a Linux graphical target permitting a NS-DK application also displaying windows to be run.

The two databases supported for this environment are Oracle (9.2 and 10G) and MySQL.

Database

The databases with native support will be:

- Oracle 9i Release 2 (9.2) and 10G (10.1)
- DB2 8.1
- Sybase 12.5.3
- MySQL 4.1 et 5.0
- SQL Server 2000 and 2005.

In addition an ODBC 3.51 driver is provided. This driver permits access to all databases with an ODBC 3 driver. In particular, it can be used to access MS-Access as a replacement alternative to NS-DBR as a basis for development or the basis of an embedded thin client.

New NS-Design interface

The principle underlying this new design is to make the most of existing screens by using the spaces available around windows during development.

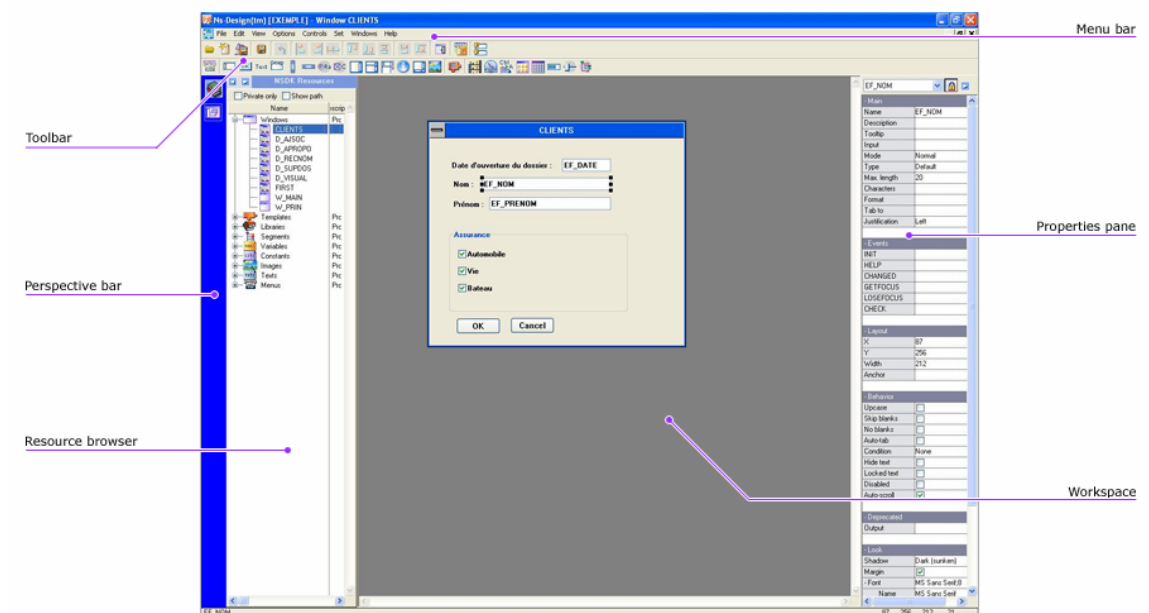
This allows:

- Easier navigation between the different resources of the reference database.
- Better visibility of the properties of the graphical controls.

New organization of the workspace

The workspace is divided into 4 major zones:

- The **menu bar** with their two tool bars at the top.
- The **resource browser** to the left.
- The graphical control **property pane** to the right.
- The **workspace** which permits the editing of windows and NCL scripts (libraries and events).



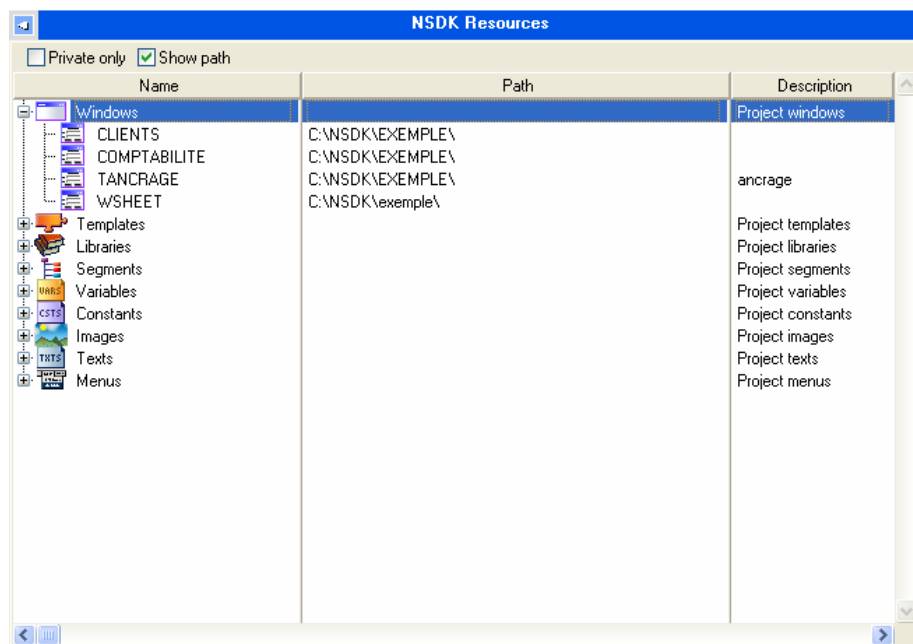
The tool becomes MDI, it is possible now to open several windows in edition but also several events and/or NCL libraries.

Resource browser

The permanently available resource browser gives a tree-structure view of your project resources. This browser can be collapsed so as to provide a full workspace.

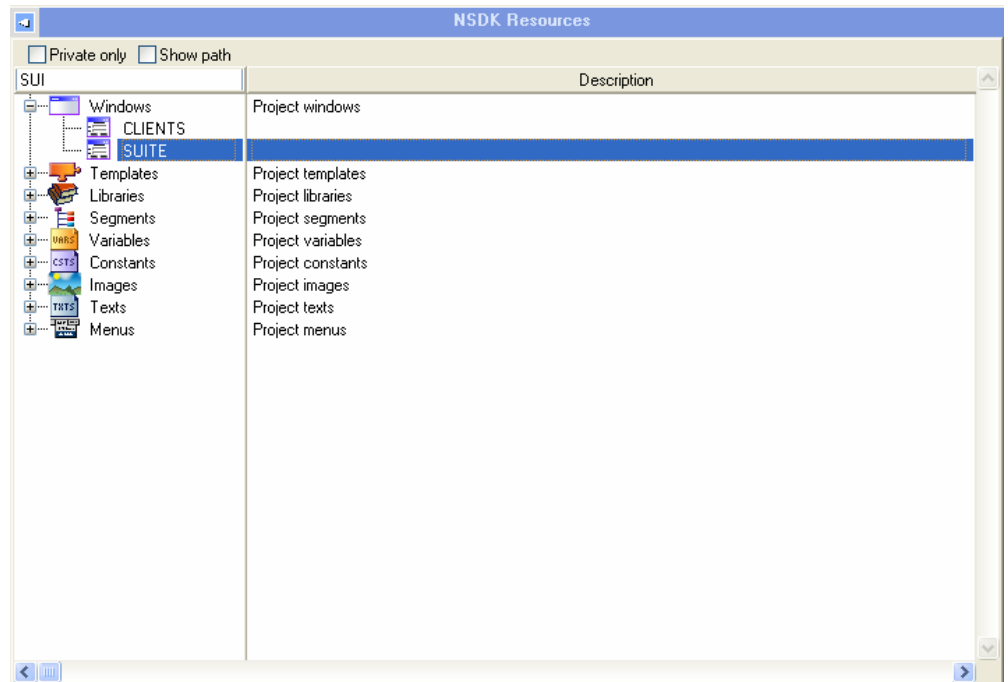
Resources are arranged according to type: The groupings available are slightly different from those in former versions of NS-DK. The list is as follows:

- NCL Libraries
- Windows
- Segments,
- Constants
- Variables,
- Images including BMP and ICO and with the addition of management for resources of the JPEG and GIF type.
- Texts
- External Menus: a new type of resource specific to this version.



It will be possible to limit the display exclusively to internal project resources (those that can be changed) by checking the box **Private only**. In addition, different availability of icons (the addition of a small padlock will permit quick identification of resources that cannot be changed).

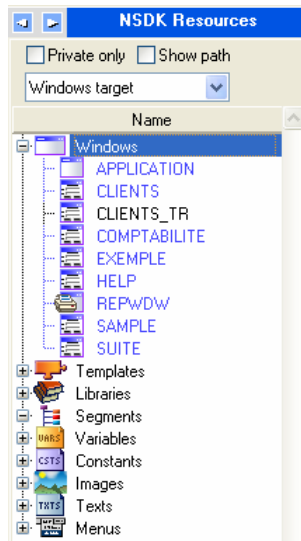
To search a resource, click on the column *Name* and activating at the same time the key [Ctrl] and enter the name.



The keys [+] and [-] of the keyboard make it possible to open or close tree structures.


Various types of images *Icons*, *Pointers* and *Bitmaps* were gathered in the type *Images*.

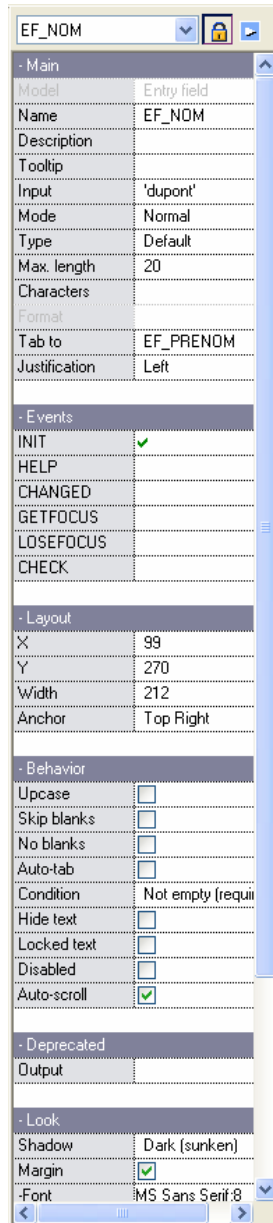
If your project is multi-target, then a list allows you in the resources browser to select resources corresponding at a specific target. Resources of other targets are displayed in blue.



The property pane

To open the property pane, select an element in the workspace (control, window...) and double-click above with the right button of the mouse.

- ◆ double-click above with the right button of the mouse
- ◆ activate the *Properties*  button in the toolbar.
- ◆ Select the *Edit / Propertie* smenu.



The **property pane** of the graphical controls is located at the right of the NS-Design workspace. It shows the different properties of the selected control. Thus, you can open it just once, and select the controls of the window in the list box at the top of the pane. The properties display by the property pane is so more effective than the *Info* boxes of each control.

The {bmc IMG00008.bmp} icon indicates that the control is locked and then you can not modify it.

The {bmc IMG00009.bmp} icon indicates that the control is not locked.

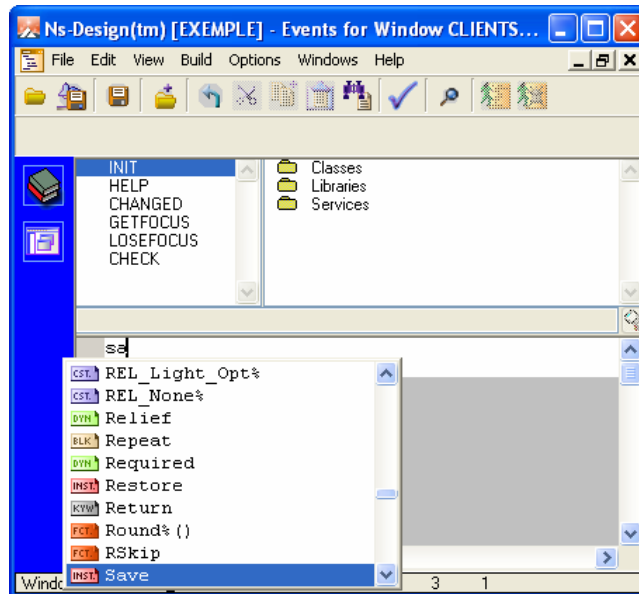
There are five main properties types: *Main*, *Events*, *Layout*, *Behavior*, *Look* :

- *Main* integrates the main properties of the control : name, description, format, justification, ...
- *Events* includes the events of the control. Those implemented have been displayed by a tick. By clicking on INIT, HELP, GETFOCUS, ... event, the *Events* window opens.
- *Layout* allows you to fix up the control in the window by indicating its vertical or horizontal position and its size in width and height. A Combo-box Editable allows you to increase or decrease quickly the size.
- *Behavior* gathered the behavioral properties: disabling the control, no focus, call mode of a window, ...
- *Look* gathered the visual look properties of the control: font, size, bold, italic, underlined, ...
- You can find other type of properties depending on the type of the selected control: *Deprecated*, *Miscellaneous*, ...

For a better properties visibility, the property pane can be resize with the mouse.

NCL editor

The NCL scripts editor (libraries and events) has been developed: it allows auto-completion of NCL verbs by pressing the [Ctrl]+[spacebar] keys.



Furthermore, auto-completion also occurs with END* instructions depending on the instructions that come before: EVALUATE, FUNCTION, IF, INSTRUCTION, LOOP, SEGMENT, EVALUATE/WHERE, WHILE, FOR, INDEXES.

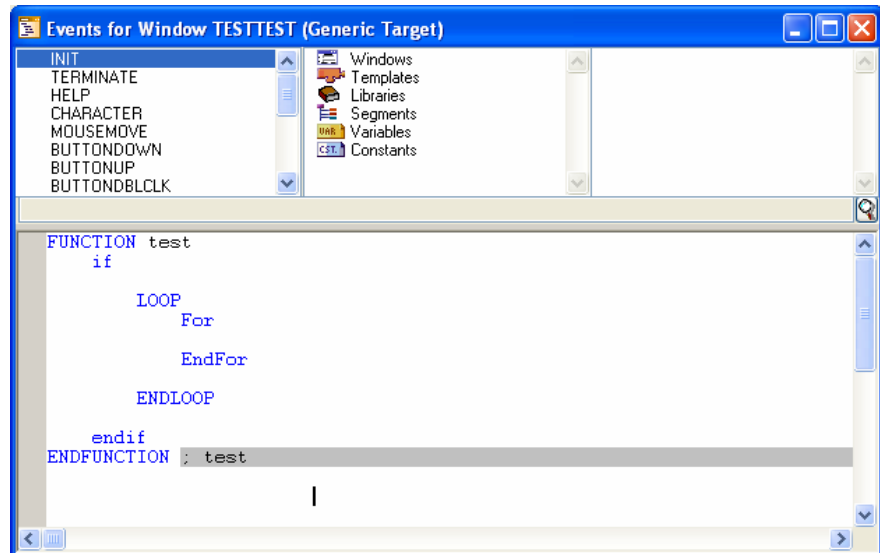


Auto-completion does not occur with the REPEAT/UNTIL block.

Auto-completion occurs when you leave the line (up/down arrows, [PageUp] key, [PageDown] key, mouse click outside the current line containing the END, [Return] key, etc).

Auto-completion of END* instructions follows the indentation and capital letters of the instruction it terminates.

ENDFUNCTION and ENDINSTRUCTION instructions are followed by a semi-colon and the name of the function/instruction.



Example:

By incorporating the following example into the NCL source editor:

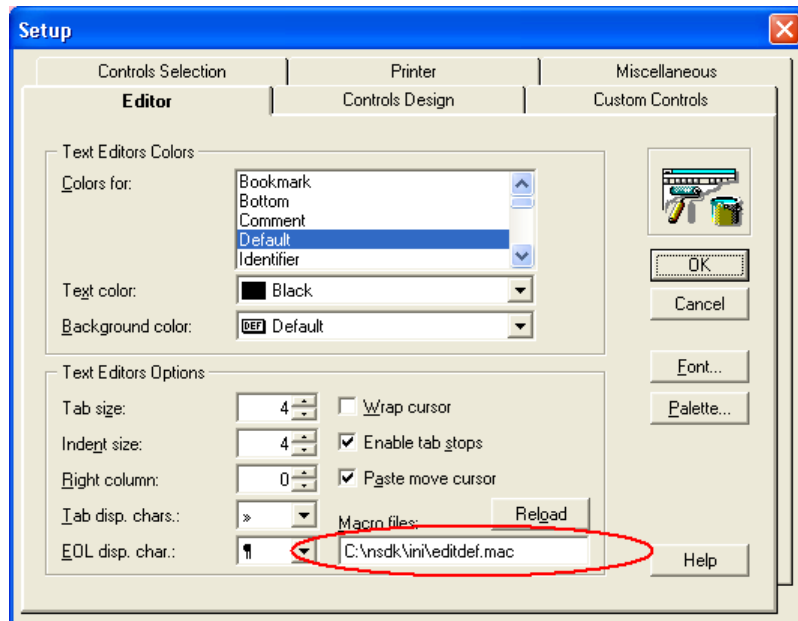
```
Function Toto$
  while
    Evaluate
      WHERE
        for
          bidule
        end
      end
    end
  end
EndFunction ; Toto$
```

Then, on leaving the line containing the last END instruction, all the END instructions are changed (if you leave a line containing an END instruction which is not the last, only the preceding ones are updated) and you obtain:

```
Function Toto$
  while
    Evaluate
      WHERE
        for
          bidule
        endfor
      ENDWHERE
    EndEvaluate
  endwhile
EndFunction ; Toto$
```

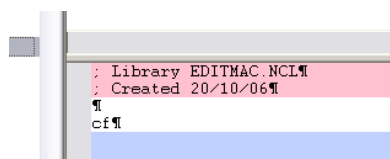
Macros

It is also possible to allow macros setup for the code sequences which are often repeated. These macros must be defined in a file of .mac extension. The path of this file is defined in the **Setup** dialog box at the **Editor** tab.



In this file several macros can be defined and must respect following syntax. An example of file macros (editdef.mac) is delivered with NS-DK (present in repertory INI at the time of the installation).

Once this file installed and its path defined on the **Option/Setup** tab, to create a heading of function for example, you enter cf in the editor:



Cursor on the end of the line and presses the keys [Ctrl]+[Shift]+[Space] and you obtain:

A screenshot of a code editor window with a light blue background. The code is written in a syntax-highlighted format with various colors (pink, blue, green, yellow) for different parts of the macro. The code defines a function named 'Function' with parameters 'Created' and 'Modified', both set to '1999'. It also includes a 'Return value' section and a 'Local variables declaration' section. The code is as follows:

```
: Library EDITMAC.NCL
: Created 20/10/06
:
:>> Function [[***]]
:>> Created [[***]]1999 by [[***]]
:>> Modified [[***]]1999 by [[***]]
:
: [[***]]
:
:>> Parameters
:>> [[***]]
:
:>> Return value
:>> [[***]]
:
:>> Local variables declaration
:
:
Things_Trace "[***] : ***** BEGIN *****"
Things_Trace "[***] : ***** Parameters In =>" & [[***]] & ";"
:
:
Things_Trace "[***] : ***** Parameters Out=>" & [[***]] & ";"
Things_Trace "[***] : ***** END *****"
```

This macro is detailed in the editdef.mac file at the line *cf.

New organization of menus and tool bars

The organization of menus has been overhauled to improve ergonomics and enhance navigation.

The File menu

The *File* menu comprises the traditional options for opening, saving and closing files, but also to stop NS-DK.

The Edit menu

The *Edit* menu comprises the traditional options of all graphic application: cut, copy, and delete. It contains also options more specific to NS-DK like the locking of published controls.

These elements of menu act on the selection of the objects (control or window) made in the workspace.

The View menu

The *View* menu allows you to navigate between the different windows of the NS-DK application and the views (*NSDK Resources*, *Opened Windows*).

The Build menu

The *Build* menu allows you to build an executable to leave the information contained in the project (resources, generation, etc...). The *Build* menu merges the old menus *Make* and *Run*.

The Options menu

The *Options* menu allows you to configure NS-DK, to manage the targets of your application, to manage the components of your services application. The menu *Project* has been withdrawn. It is merged with the *Options* menu now.

The Controls menu

The menu *Controls* has been divided into three parts (*Standard*, *Win32* and *Custom*) to facilitate the use of controls.

New keyboard shortcuts

The following tables summaries the new keyboard shortcuts allowing you to carry out some tasks more quickly.

Keyboard shortcuts of the File menu

Keyboard shortcuts	Description
[Ctrl]+S	Save the current window.
[Shift]+[F2]	Save all the opened windows
[Ctrl]+[F4]	Close the current window
[Alt]+[F4]	Close NS-DK

Keyboard shortcuts of the Edit menu

Keyboard shortcuts	Description
[Alt]+[Backspace]	Cancel the last action.
[Alt]+[Enter]	Redo the last action.

Keyboard shortcuts	Description
[Shift]+[Del]	Cut the selected item
[Ctrl]+[Ins]	Copy the selected item
[Shift]+[Ins]	Past the selected item
[Del]	Erase the selection.
[F9]	Arrange the selected controls between them.
[Shift]+[F7]	(Un)Lock the current window without modifyng the lock of the controls belonging to the window.
[Ctrl]+[F7]	(Un)lock all controls of the current window, without modifyng the lock of the window.
[F7]	Allow to switch between the lock and unlock of the selected items.

Keyboard shortcuts of the View menu

Keyboard shortcuts	Description
[Ctrl]+[Alt]+L	Open the <i>Libraries</i> view.
[Ctrl]+[Alt]+W	Open the <i>Opened Windows</i> view.
[Alt]+[F8]	Open the <i>Targets Info</i> window which lists the targets of the current window.
[Shift]+[Enter]	Open the script editor.

Keyboard shortcuts of the View menu

Keyboard shortcuts	Description
[Ctrl]+[Shift]+B	Generation of the source files.
[F5]	Launches the NS-DK project.

Actions of copy/paste

Keyboard shortcuts	Description
[Alt]+[Backspace] ou [Ctrl]+[Shift]+[Z]	Cancel an action
[Alt]+[Enter] ou [Ctrl]+[Shift]+[A]	Redo an action.
[Ctrl]+[X] ou [Shift]+Del	Cut
[Ctrl]+[C] ou [Ctrl]+[Ins]	Copy
[Ctrl]+[V] ou [Shift]+[Ins]	Past
[Ctrl]+[Shift]+[Del]	Cut in addition at the end of the text already present in paperweight
[Ctrl]+[Shift]+[Ins]	Copy in addition at the end of the text already present in paperweight
[Ctrl]+[Shift]+[M]	To switch between the mode where the Copy moves the cursor with the mode when the cursor remains at the beginning of the paste text.

Other manipulations in the code editor

Keyboard shortcuts	Description
[F8]	Verification of the code
[Tab]/[Shift]+[Tab]	Indentation in front of /back of the selection
[Alt]+[Curseur] [Alt]+[Shift]+flèches	Rectangular selection
[Ctrl]+[F]	Open the Find box
[Ctrl]+[Shift]+[R] ou [Alt]+[Shift]+[F9]	Open the Replace box
[F9]	Search the next occurrence
[Shift]+[F9]	Search the previous occurrence

Keyboard shortcuts	Description
[Ctrl]+[F9] et [Ctrl]+[Shift]+[F9]	Search the selected text at the bottom ([Ctrl]+[F9]) or at the top ([Ctrl]+[Shift]+[F9]) at the code.
[Ctrl]+[Shift]+[G]	Open the Goto box
[Ctrl]+[Shift]+[U]	Conversion in upper case
[Ctrl]+[Shift]+[L]	Conversion in lower case
[Ctrl]+[F12]	Add/remove the bookmarks on the lines
[F12]	Next bookmark
[Shift]+[F12]	Previous bookmark
[Alt]+[F12]	First bookmark
[Alt]+[Shift]+[F12]	Last bookmark
[Ctrl]+[Shift]+[F12]	Remove all the bookmarks
[Ctrl]+[Enter]	Cut a line without any indentation
[Ctrl]+[ArrowUp] et [Ctrl]+[ArrowDown]	Scrolling the text by taking the cursor in the same line.
[Ctrl]+[Shift]+[Enter]	Search the same occurrences of type [***]
[Ctrl]+[Shift]+[BackSpace]	Extension of the macros texts
[Ctrl]+[PageUp] et [Ctrl]+[PageDown]	Navigation between the items of the same block of NCL code
[Shift]+[F1]	Search the selected item in the NCL libraries and display its syntax
[Ctrl]+[F1]	Display the online help corresponding at the selected item
[Shift]+[F5]	Reach out the code of a user instruction or function.

Modification of some dialog boxes

The dialog boxes allowing the definition of the following resources have been modified:

- Constant,
- Segment,
- Variable.

The *Edit Segment* and *Edit Variable* boxes contain a *Type* field allowing you to define the segment or variable type. Besides, the fields of the segment/variables can be declared by reference (*Reference* field) to be used by casted pointers.

Edit segment RGBCOLOR

Segment
Name: RGBCOLOR
Description: Created 06/02/06

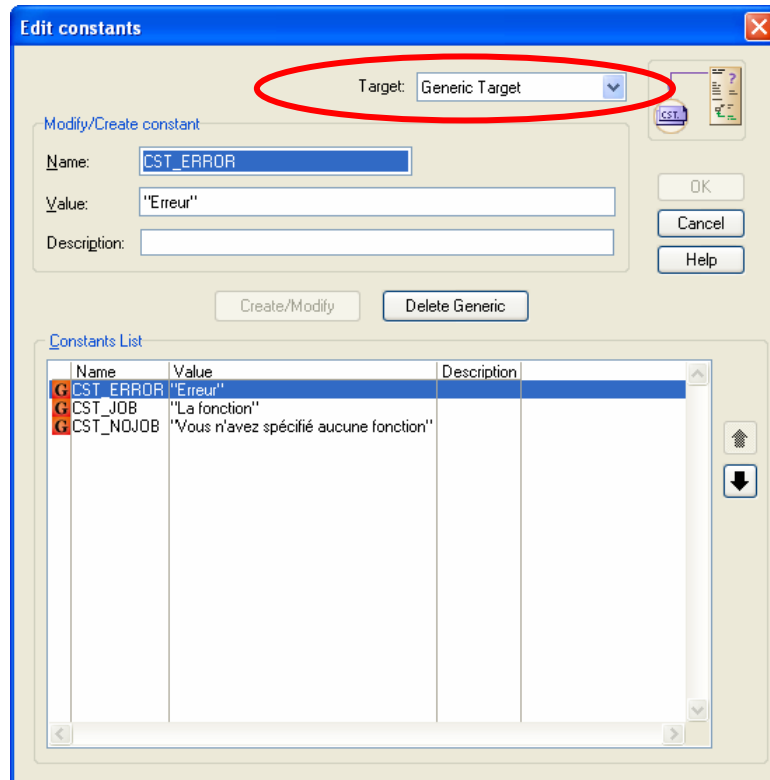
Fields details
Name: RED%
Type: INT
Size: 1
Dimensions: 128
Description: ☐ Reference

Fields List

Type	Name	Size	Dims	Description
INT	RED%	(1)	[128]	
INT	BLUE%	(1)	[20]	
INT	GREEN%	(1)	[]	

Buttons: OK, Cancel, Indexes..., Help, Append, Insert, Modify, Delete

The *Edit Constants* box allowing you to select the constants depending on the target.



The dialog boxes allowing the tool and projects configuration have been modified as well.

Deleting historical limitations

NS-DK has several number of limitations which are the result of the Windows and OS/2 16-bit era.

XNP file

The main file responsible for these limits is the project description file (.PRJ) and its binary format. In order to respond to requests from clients and to remove these limits, NS-DK projects (.PRJ file) will switch to XML format and the .XNP file.

The .XNP format contains the files of project PRJ and of configuration N_C. The XML format of XNP file makes it possible to store several N_C files.

A N_C file contains the parameters of generation so much for source files generated by NS-Gen that for the compiler and data link with to use to obtain achievable.

Contrary to the PRJ files which are binary files, the XNP files are text files and thus can be read with an editor of text.

Size of the resource name and new resources

The switch to XML format for .PRJ files will permit:

- A move to 220 characters for the names of directories used for storing resources.
- A move to 31 characters for the names of SCRs, NCLs and images.
- The option of managing GIF and JPEG resources in addition to BMPs.

64 KB limitations

The 64 KB limitation for NCL (libraries) and SCR (screen) type resources will be removed.

Improvement of the NCL language

The NCL language is going to evolve with the appearance of new APIs, and also new types.

New type : DynStr

This new type will permit the definition of character strings without their size having to be defined. The size of the variable will be adapted automatically according to its contents.

The size of the contents is not limited to 255 characters.

The conversion of a DynStr type variable into a CString will be automatic and vice versa.

DynStr will therefore be usable with functions/instructions taking CString as a parameter. However, some APIs have been optimized for DynStr.

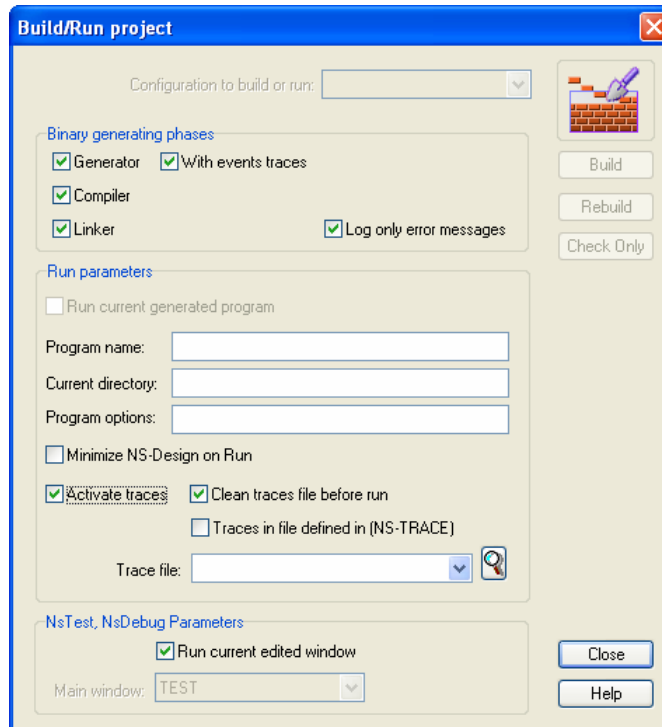
Extension of the POS% function

The POS% function has improved. It returns the position of all occurrences of a character string in another character string.

Trace API

A trace API has been added. It allows you to trace simply events and functions. The trace is enabled or disabled according to how an environment variable is set.

To generate trace for the code C, tick the box *With traces event* in the *Build/Run project* dialog box. Trace is generated in a text file.



Tick the *Activate traces* box, to obtain the traces with the binary execution, in the file indicated by the variable of environment NS-TRACE, that is to say in the file selected in Trace file.

The user can add these proper traces in his programs NCL by the call of instruction NS_TRACE.

The SetNSGenTrace instruction allows you to modify the formatting of traces.



For more information on the functions and instructions, please refer to the NSMISC library at the Traces management, or to have an outline in the chapter 2 of this manual.

Changes to NCL services

A great number of NS-DK NCL services have parameters defined as INT(4), while it is actually a question of POINTERS.

Version 5.00 fixes this problem so to improve future developments.

For the time being the behaviour remains unchanged for Windows 32-bit applications.

Integer and pointer types sizes in the NCL language are different according to the platform, 32 or 64-bit.

Type of identifier	Size in 32 bits		Size in 64 bits	
	Bytes	Bits	Bytes	Bits
POINTER	4	32	8	64
INT(4)	4	32	4	32
INT(2)	2	16	2	16

In a 32-bit system, the INT(4) and POINTER types are the same size. It is therefore possible to substitute one for the other.

In a 64-bit system, the INT(4) type is **no longer** compatible with the POINTER pointer so assignment can no longer be made from one to the other.



For more information, please refer to the "NCL language" manual or the chapter 3 of this manual.

Passing parameters to an event

Passing a parameter bigger than 2 bytes to an event requires it to be broken down into two, 2 byte, words prior to passing it, via the Hiw and Low instructions.

With version 5.00 this prior breakdown will no longer be necessary; it will be possible to use the syntax:

SEND EXECUTED POINTER TO CONTROLNAME

Instead of:

SEND EXECUTED Low POINTER, Hiw POINTER TO CONTROLNAME

This change of syntax is also valid for the instructions POST, OPEN, CALL, etc.



For more information, please refer to the "NCL language" manual.

Enhancement of NSWIN

The Windows specific service NSWIN.NCL has been extended. The following functionalities have been added:

- Management of the Windows Registry APIs.

- Management of the mouse pointer position APIs have been added permitting the the mouse position recuperation under Windows.



For more information, please refer to the chapter 5 of "Services Libraries Reference-Volume 1", manual or to the chapter 2 of this manual.

Management of the Windows Registry

The Windows system saves the data corresponding to its configuration in a data base called the Registry.

The Registry keeps the profiles of each user of the computer as well as the data of the system peripherals, the properties of the installed programs.

A Registry editor is provided with Windows called regedit.exe.

The Registry API in the NS-WIN service allows to control and to modify the Windows Registry. However, these functions are exclusively for experienced users. It is not recommended to modify the base of Windows Registry. Nat System recommends that you leave the Windows programs modify themselves the Registry.

The use of these functions is particularly delicate. In fact, a bad modification of the Windows Registry can damage seriously your system. Nat System recommends to you all modification to save your critical data. If despite everything, your system is damaged often the only remaining option is to reinstall your system. For more information about the Windows Registry and the possible problems due to its modification, please refer to the online help of Windows.

Management of the mouse pointer position

NS-DK integrates two new instructions: GETCURSORPOS and SETCURSORPOS respectively allowing to recuperate the current position of cursor and to position the cursor with the position requested.



For more information on these new instructions, please refer to the chapter 2 of this manual.

Graphical controls and display

NS-DK 5.00 brings many enhancements for the graphical controls. Graphical representation of Nat System's application was also improved.

- Controls anchoring
- NS-ScreenExploser allows you to modify the controls character size.
- List control
- External menu control

Anchoring controls

Anchoring controls means that during redimensioning of the window they are able to adapt to the new size of the window and take up their appropriate position and size.

For example, if a Push Button control is anchored to the top left edge of the window, it is positioned in such a way that it will always remain the same distance from the left and top edges of the window.



The **Size Redraw** option in the properties pane of the window must be enabled in order to reposition/resize controls each time the user changes the size of the window.

Principle

Each control has four sides that can be anchored to the edge of the window in three ways (total = $3^4 = 81$ anchoring options per control).

- **Fixed** anchoring lets you keep a constant distance between the control and the edge of the window. If the window is redimensioned, the distance between the control and the edge of the window remains the same.
- **Proportional** anchoring lets you keep the proportion from the control to the edge of the window constant as compared with the size of the window. If the window is redimensioned, the distance between the side of the control and the edge is always proportional to the size of the window.
- **No anchoring.** The side of the control is not attached to the edge of the window.

If a control is not anchored on two opposite sides, when the window is redimensioned, the position of its centre remains in proportion as compared with the edges of the window.



Some controls or control attributes cannot be resized: the height of Entry-Field, Static-Text, Radio buttons, the width of controls in auto-size mode etc.

You can anchor controls in two ways:

- ◆ Using the **Anchor** parameter in the control properties pane (which lets you choose between nine of the most common types of anchoring).

NS_AS_BOTTOM_LEFT	Anchoring to the bottom left (default value)
NS_AS_TOP_LEFT	Anchoring to the top left.
NS_AS_TOP_RIGHT	Anchoring to the top right.
NS_AS_BOTTOM_RIGHT	Anchoring to the bottom right.
NS_AS_WIDTH_BOTTOM	Anchoring to the bottom left, right.
NS_AS_WIDTH_TOP	Anchoring to the top left, right.
NS_AS_HEIGHT_LEFT	Anchoring to the left bottom, top.
NS_AS_HEIGHT_RIGHT	Anchoring to the right bottom, top.
NS_AS_WIDTH_HEIGHT	Anchoring to the left and right, bottom, top.
- ◆ Using the dynamic .ANCHOR parameter (in write mode only) with the NS_AS_* constants set in the NSMisc the library, which lets you choose between all the anchoring options.
 - NS_AS_LEFT
 - NS_AS_RIGHT
 - NS_AS_TOP
 - NS_AS_BOTTOM
 - NS_AS_PLEFT
 - NS_AS_PRIGHT
 - NS_AS_PTOP
 - NS_AS_PBOTTOM
 - NS_AS_NONE
 - NS_AS_BOTTOM_LEFT
 - NS_AS_TOP_LEFT

- NS_AS_BOTTOM_RIGHT
- NS_AS_TOP_RIGHT
- NS_AS_WIDTH_TOP
- NS_AS_WIDTH_BOTTOM
- NS_AS_WIDTH_HEIGHT
- NS_AS_HEIGHT_LEFT
- NS_AS_HEIGHT_RIGHT

Examples of use

The control is anchored to the bottom left, its size is fixed.

```
CTRL.ANCHOR=NS_AS_BOTTOM_LEFT
```

The control is anchored to the top right, its size is fixed.

```
CTRL.ANCHOR=NS_AS_TOP_RIGHT
```

The control is anchored to the left, to the right and at the bottom, its width changes with the window.

```
CTRL.ANCHOR=NS_AS_WITH_BOTTOM
```

The control is anchored to the left, to the right, at the top and at the bottom, its width/height changes with the window. This mode is generally applied to list boxes.

```
CTRL.ANCHOR=NS_AS_WITH_HEIGHT
```

The control is anchored at the bottom and to the left (fixed) and to the right (in proportion), the position of its right edge is proportional to the size of the window.

```
CTRL.ANCHOR=NS_AS_LEFT & NS_AS_RIGHT & NS_AS_BOTTOM
```

The control is anchored at the bottom (fixed), to the left and to the right (in proportion), the positions of its right and left edges, as well as its width, are proportional to the width of the window.

```
CTRL.ANCHOR=NS_AS_PLEFT & NS_AS_PRIGHT & NS_AS_BOTTOM
```

Warning, a special case: this control is anchored at the bottom, but neither to the right or the left. In this case the width of the control is fixed and its centre always occupies the same position in proportion as compared with the left and right edges of the window.

```
CTRL.ANCHOR=NS_AS_BOTTOM
```

Use

Anchoring can mainly be used to solve two problems.

1. To make applications re-sizable and repositionable.
2. To create applications that automatically adapt to the screen resolution.

Development recommendations

Example: To write an application that automatically adapts to the screen resolution (800/600 or 1024/768):

- The developer draws his windows in 800/600 and changes the control anchoring (using the properties pane or by the ANCHOR dynamic qualifier.
- He maximizes his window in the INIT event or via the design. If anchoring is coded well, the look of the window will always be correct and the volume of information displayed does not depend on screen resolution.

Dynamic changes

At any place in the application, the positions/sizes of controls, as well as their anchoring, can be changed dynamically, updating being automatic.

See also

You can associate anchoring with the SetAttachedChildFlags instruction in order to create windows with functionalities and a modern look.



For more information on the SetAttachedChildFlags instruction, please refer to the "Service Libraries Reference - Volume 1 - GUI and Printing APIs" manual.

Comment

When a window becomes small and in order to avoid overlapping of controls, it may be useful to force the minimum dimension of a window using the AdjustSizeOrPos event.

This event is generated when the user wants to change the position or size of a window. It lets you intercept this request and change the values of the new positions and sizes via programming.



For more information on the AdjustSizeOrPos event, please refer to the "NCL Language" manual.

NS-ScreenExploser

This module allows by the configuration of NSLIB.INI file to change the size of the characters of various NS-DK controls: controls like CheckBox or RadioButton which do not change size with anchoring are impacted by ScreenExploser.

SheetBox control

The new SheetBox control aims to better replace the old List Box control. SheetBoxes do not have the 255 characters limit.

This control changes with version 5.00: use its property pane to set

- Title colours
- The colour of even and odd lines
- The colour of even and odd columns.
- To anchor one or more columns to the left.
- To define an on-line entry format for each column.

Moreover, it will be possible to set dynamically:

- The colour of a cell.
- The Drag & Drop feature.
- The column sort feature.



For more information, please refer to the "NSSHTREE SheetBox library" chapter of the "Services Libraries Reference - Volume 1".

External menu control

A new resource will appear in NS-DK 5.00: an external menu.

The advantages of dynamic Menu resource management are the following:

- You can associate a single menu bar definition with several different windows. **Menu resources are re-usable**, because they are independent of any particular window.
- Without specific programming, you can make available a **tool bar that's synchronous with menu items**; in other words, the tool bar displays buttons that always correspond to the items on the menu and trigger the same processes.

- You can **override** a parent window's menu bar with a child window's menu bar, without duplicating the parent window's menu items, and thus manage **MDIs** (Multiple Document Interface) transparently.



Don't confuse this form of menu management with designing Menu controls. Menu controls let you design totally **static** menus. They are an integral part of the definition of the window they were designed for. If you want an associated tool bar, you must program it separately.

The NSDKCFG.INI and NSDKLOC.INI files

The NS-DK.INI file, which allows you to provide a whole of tools and parameters suggested by default in the various generation dialog boxes, is obsolete. It is from now on replaced by two files:

- NSDKCFG.INI which corresponds to the global configuration of your projects.

The NSDKCFG.INI file is delivered with NS-DK. It has the role to be divided on an NS-DK installation in a network for example. The NSDKCFG.INI file is normally modified by one administrator of NS-DK installation or a project manager.

- NSDKLOC.INI which corresponds to the local configuration of a project.

The NSDKLOC.INI file is not provided with NS-DK. It is a configuration file suitable for each user, who creates it and modifies it.

Nat System suggests you to point the NS-INI environment variable on a user local directory, with one NS-DK installation possibly on a network.

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

The NSDKCFG.INI file is firstly required in the following directories:

1. In the directory pointed by the NS-INI environment variable (thus without the name of the project),
2. If there does not exist in the directory pointed by the NS-INI environment variable, it is required in the */ini* directory of the installation.
3. If there does not exist in the */ini* directory, it is created in the directory pointed by NS-INI.
4. If the directory pointed by the NS-INI environment variable does not exist or if NS-INI is not defined, it is created in the */ini* directory installation.

The NSDKLOC.INI file is firstly required in the directory pointed by NS-INI:

1. If there does not exist in the repertory pointed by the variable of environment NS-INI, it is created in the repertory pointed by NS-INI.
2. If the repertory pointed by NS-INI does not exist or if NS-INI is not defined, it is required in repertory INI of the installation.
3. If there does not exist in repertory INI of the installation, it is created in repertory INI of the installation.



If the NS-DK installation is divided in network, and if the NS-INI variable is not defined for several users, there is a risk of collision on the NSDKLOC.INI file which becomes shared (there is no protection for the simultaneous writing by several users). On the contrary, if the installation of NS-DK is distributed on each station, and if the variable NS-INI is not defined, each user will create a NSDKLOC.INI file in the */ini* directory of the local installation of NS-DK.

Generation and debugging

NS-DK provides different solutions for the generating and debugging applications.

- Automatic trace generator
- Optimization of the generator to support 64 bits targets.
- Size of the DLLs names

Limitation

It is not from now on possible any more to have a generation configuration allowing the generation of several targets.

Consequently, during the migration of an application developed in preceding versions and containing several targets, only the default target is recovered.

Automatic trace generator

NS-DK 5.00 provides developers with a new generation option which will permit automatic generation of traces during a call to a function/instruction and its exit, as well as the entry and exit of an event.

This functionality will be based on the new API of trace. Traces will be indeed active only at the time of positioning of the good environment variable.

NS-DebugMem

This utility will be provided in version 5.00. It lets you detect memory leaks during execution of an NS-DK application.

Memory leaks are detected independently of their impact on execution. it will be possible to correct stability problems simply.

Optimization for 64-bits

Generation is changed so as to become compatible with 64-bit targets.

Some automatic transtyping mechanisms taken care of by the generator have thus been improved.

DLL names

The size of the DLL name generated is no longer limited to 8 characters.

Chapter 2

New fonctions and instructions



This chapter presents the new functions and instructions introduced in NS-DK 5.00 for development optimization.

***You will find in
this chapter***

- The new functions and instructions of the NSWIN and NSMISC libraries.
- The new NSSEQ library.

Table of contents

NSWIN library	2-3
Reading/Writing in the Windows Registry 2-3	
Managing the mouse pointer position 2-3	
Attachement of windows 2-3	
NSMisc library	2-19
Managing traces 2-19	
Managing the anchoring controls 2-19	
NSSEQ library	2-28

NSWIN library

For recall, this library provides Windows-specific functions, such as enumerating open windows or sending messages.

This library is intended for programmers with a very sound knowledge of the internal Windows mechanisms for handling events and windows.

Reading/Writing in the Windows Registry

CREATEMODIFYREGISTRYKEY%	2-4
READREGISTRYKEY%	2-6
DELETEREGISTRYKEY%	2-8
DELETEREGISTRYVALUE%	2-9
NS_HKEY_ *%	2-10
NS_REG_ *%	2-11

Managing the mouse pointer position

GETCURSORPOS	2-12
SETCURSORPOS	2-13

Attachement of windows

SetAttachedChildFlags	2-14
GetAttachedChildFlags%	2-15
ACF_ *%	2-16

CREATEMODIFYREGISTRYKEY% function

Allows you to create and modify the key of the Windows Registry.

Syntax **CREATEMODIFYREGISTRYKEY%** (*Hkey%*, *Supkey\$*, *valueName\$*, *Type%*, *Value*)

Parameters	<i>Hkey%</i>	INT(4)	I	value of the predefined key (NS_HKEY_%% constants)
	<i>Supkey\$</i>	CSTRING	I	full path name of the key
	<i>valueName\$</i>	CSTRING	I	key name
	<i>Type%</i>	INT(4)	I	data type (NS_REG_%% constants)
	<i>Value</i>	CSTRING	I	key value

Returned value INT

value	meaning
NS_ERROR_SUCCESS%	creation or modification proceeds correctly
NS_ERROR_CANTOPEN%	key does not exist
NS_ERROR_CANTWRITE%	a problem emerges during the writing of the value
NS_ERROR_INVALID_PARAMETER%	If the key type is different from NS_REG_SZ%, NS_REG_EXPAND_SZ%, NS_REG_DWORD%, the creation or the modification can not happen

Notes

1. The key value (Value parameter) is defined by a CSTRING but can be a CSTRING of n size or a dynamic STRING.
2. When the key does not exist, the function creates the key. If the key exists, the function modifies the key.
3. Currently, Nat System proposes the three most used main data types (see NS_REG_%% constants). The other data types (REG_BINARY, REG_DWORD_BIG_ENDIAN, REG_LINK, REG_MULTI_SZ, REG_RESOURCE_LIST, REG_NONE) have not been implemented yet. The function returns NS_ERROR_INVALID_PARAMETER if you use a non-implemented type.

4. In the case of an integer, the entering values are - 2147483647 à 2147483647 (32 bits integer).
5. Because of the Windows limitations, it is not possible to create a new directory directly under the following keys roots:
NS_HKEY_CURRENT_USER%
NS_HKEY_LOCAL_MACHINE%
In that case, the function returns NS_ERROR_INVALID_PARAMETER%.

Example

```
*****  
;*** Key creation  
*****  
  
Local s1$,s2$  
Local Dynstr s3  
Local int l_ret%  
  
s1$ = «Network\Z1»  
s2$ = «DeferFlags»  
s3 = «Test 255 lower length»  
  
l_ret% = CREATEMODIFYREGISTRYKEY%  
(NS_HKEY_CURRENT_USER,s1$,s2$,s3,NS_REG_SZ)  
Evaluate l_ret%  
Where NS_ERROR_SUCCESS%  
  Message "Creation/Modification","OK"  
EndWhere  
Where NS_ERROR_INVALID_PARAMETER%  
  Message "Creation/Modification","Invalid parameter"  
EndWhere  
Where NS_ERROR_CANTWRITE%  
  Message "Creation/Modification","Impossible writing"  
EndWhere  
EndEvaluate
```

See also

READREGISTRYKEY%, DELETEREGISTRYKEY%,
DELETEREGISTRYVALUE%, NS_HKEY_%% and NS_REG_%% constants

READREGISTRYKEY% function

Allow to read a key of the Windows Registry.

Syntax **READREGISTRYKEY%** (*Hkey%*, *Supkey\$*, *valueName\$*, *Type%*, *Value*)

Parameters	<i>Hkey%</i>	INT(4)	I	value of the predefined key (NS_HKEY_%% constants)
	<i>Supkey\$</i>	CSTRING	I	full path name of the key
	<i>valueName\$</i>	CSTRING	I	key name
	<i>Type%</i>	INT(4)	O	data type filled by the function (NS_REG_%% constants)
	<i>Value</i>	DYNSTR	O	key value

Return value INT

value	meaning
NS_ERROR_SUCCESS%	creation or modification proceeds correctly
NS_ERROR_CANTOPEN%	the key does not exist
NS_ERROR_CANTREAD%	a problem emerges during the reading of the value
NS_ERROR_UNKNOWNNTYPE%	If the key type is different from NS_REG_SZ%, NS_REG_EXPAND_SZ%, NS_REG_DWORD%, the reading can not be happen

Notes

1. The key value (Value parameter) is defined by a CSTRING but can be a CSTRING of n size or a dynamic STRING.
2. The Value parameter allows you to read the strings of more than 255 characters.
3. Currently, Nat System proposes the three most used main data types (see NS_REG_%% constants). The other data types (REG_BINARY, REG_DWORD_BIG_ENDIAN, REG_LINK, REG_MULTI_SZ, REG_RESOURCE_LIST, REG_NONE) have not been implemented yet. The function returns NS_ERROR_UNKNOWNNTYPE% if you use a non-implemented type.

Example

```
*****  
;*** Key reading  
*****  
  
Local s1$,s2$  
Local Dynstr s3  
Local Type%  
Local int l_ret%  
  
s1$ = «Network\Z1»  
s2$ = «DeferFlags»  
  
l_ret% = READREGISTRYKEY% (NS_HKEY_CURRENT_USER,s1$,s2$,type%,s3)  
  
Evaluate l_ret%  
  
Where NS_ERROR_SUCCESS%  
    Message "Reading","Type key"&& type%&&"and of value"&&s3  
EndWhere  
  
Where NS_ERROR_CANTOPEN%  
    Message "Reading","non-existent key"  
EndWhere  
  
Where NS_ERROR_CANTREAD%  
    Message "Reading","Impossible reading"  
EndWhere  
  
Where NS_ERROR_UNKNOWN_TYPE%  
    Message "Reading","Reading of this type value impossible "  
EndWhere  
  
EndEvaluate
```

See also

CREATEMODIFYREGISTYKEY%, DELETEREGISTRYKEY%,
DELETEREGISTRYVALUE%, NS_HKEY_%% and NS_REG_%% constants

DELETEREGISTRYKEY% function

Allows you to delete all keys of a directory in the Windows Registry.

Syntax **DELETEREGISTRYKEY%** (*Hkey%*, *Supkey\$*)

Parameters	<i>Hkey%</i>	INT(4)	I	value of the predefined key (NS_HKEY_%% constants)
	<i>Supkey\$</i>	CSTRING	I	full path name of the key

Return value INT

Value	meaning
NS_ERROR_SUCCESS%	creation or modification proceeds correctly
NS_ERROR_CANTOPEN%	the key does not exist
NS_ERROR_CANTDELETE%	a problem emerges during the cancellation

Note

1. The cancellation of a key directory containing itself sub-directories is not possible. It is necessary to delete the sub-directories first and then the directory.

See also CREATEMODIFYREGISTRYKEY%, READREGISTRYKEY%,
DELETEREGISTRYVALUE%, NS_HKEY_%% constants

DELETEREGISTRYVALUE% function

Allows you to delete a key (identified by the *valueName\$* parameter) of the Windows Registry..

Syntax **DELETEREGISTRYVALUE%** (*Hkey%*, *Supkey\$*, *valueName\$*)

Parameters	<i>Hkey%</i>	INT(4)	I	value of the predefined key (NS_HKEY_ *% constants)
	<i>Supkey\$</i>	CSTRING	I	full path name of the key
	<i>valueName\$</i>	CSTRING	I	key name

Return value INT

value	meaning
NS_ERROR_SUCCESS%	creation or modification proceeds correctly
NS_ERROR_CANTOPEN%	the key does not exist
NS_ERROR_CANTDELETE%	a problem emerges during the cancellation

Example

```
*****
;*** Cancellation of a key value
*****
Local s1$,s2$
Local int l_ret%
s1$ = «Network\Z1»
s2$ = «DeferFlags»

l_ret% = DELETEREGISTRYVALUE% (NS_HKEY_CURRENT_USER,s1$,s2$)

Evaluate l_ret%
  Where NS_ERROR_SUCCESS%
    Message "Cancellation", "OK"
  EndWhere
  Where NS_ERROR_CANTOPEN%
    Message "Cancellation", "Non-existent key"

EndWhere
  Where NS_ERROR_CANTDELETE%
    Message "Cancellation", "Impossible Cancellation"
  endwhile
EndEvaluate
```

See also CREATEMODIFYREGISTRYKEY%, READREGISTRYKEY%,
DELETEREGISTRYKEY%, NS_HKEY_ *% constants

NS_HKEY_*% Constants

These constants correspond to the predefined keys in the Windows Registry. They are used by the CREATEMODIFYREGISTRYKEY%, READREGISTRYKEY%, DELETEREGISTRYKEY% and DELETEREGISTRYVALUE% functions. .

Syntax

NS_HKEY_CLASSES_ROOT%
NS_HKEY_CURRENT_USER%
NS_HKEY_LOCAL_MACHINE%
NS_HKEY_USERS%
NS_HKEY_PERFORMANCE_DATA%
NS_HKEY_CURRENT_CONFIG%
NS_HKEY_DYN_DATA%

Notes

1. These constants correspond to the predefined keys in the Windows Registry: HKEY_CLASSES_ROOT, HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, HKEY_USERS, HKEY_CURRENT_CONFIG, HKEY_PERFORMANCE_DATA, HKEY_CURRENT_CONFIG, HKEY_DYN_DATA.



For more information about the predefined keys, see the Windows Registry documentation.

2. The internal declaration of these constants is:

CONST NS_HKEY_CLASSES_ROOT%	\$80000
CONST NS_HKEY_CURRENT_USER%	\$80001
CONST NS_HKEY_LOCAL_MACHINE%	\$80002
CONST NS_HKEY_USERS%	\$80003
CONST NS_HKEY_PERFORMANCE_DATA%	\$80004
CONST NS_HKEY_CURRENT_CONFIG%	\$80005
CONST NS_HKEY_DYN_DATA%	\$80006

See also

CREATEMODIFYREGISTYKEY%, READREGISTRYKEY%,
DELETEREGISTRYKEY%, DELETEREGISTRYVALUE%, NS_REG_*%
constants

NS_REG_.* Constants

These constants correspond to the defined data types. They are used by the `CREATEMODIFYREGISTRYKEY%` and `READREGISTRYKEY%` functions.

Syntax

`NS_REG_SZ%`
`NS_REG_EXPAND_SZ%`
`NS_REG_DWORD%`

Notes

1. These constants correspond to the defined data types of the Windows Registry: `REG_SZ`, `REG_EXPAND_SZ` and `REG_DWORD`.
2. Currently, Nat System proposes the three most used main data types. The other data types (`REG_BINARY`, `REG_DWORD_BIG_ENDIAN`, `REG_LINK`, `REG_MULTI_SZ`, `REG_RESOURCE_LIST`, `REG_NONE`) have not been implemented yet.
3. Their signification are the following:
`NS_REG_SZ%` String of steady length
`NS_REG_EXPAND_SZ%` String of variable length. This data type contains the environment variables as `%PATH%`.
`NS_REG_DWORD%` Specify a 32 bits number.



For more information about the data types, see the Windows Registry documentation.

The internal declaration of these constants is:

<code>CONST NS_REG_SZ%</code>	1
<code>CONST NS_REG_EXPAND_SZ%</code>	2
<code>CONST NS_REG_DWORD%</code>	4

See also `CREATEMODIFYREGISTRYKEY%`, `READREGISTRYKEY%`, `NS_HKEY_.*%` constants

GETCURSORPOS instruction

Allows you to retrieve the current position of the cursor.

Syntax **GETCURSORPOS** *handle%*, *x%*, *y%*

Parameters	<i>handle%</i>	POINTER	I	handle of the window
	<i>x%</i>	INT(4)	O	position in x-axis
	<i>y%</i>	INT(4)	O	position in y-axis

Note

1. The *handle%* parameter could be:
 - ◆ The handle of the window (in this case the 0,0 is the left bottom corner of the NS-DK window),
 - ◆ the handle of a control (SELFFROMCONTROL%) (in this case the 0,0 is the left bottom corner of the control)
 - ◆ 0 and in this case the 0,0 is at the left bottom of the screen.

See also SETCURSORPOS

SETCURSORPOS instruction

Sets the cursor to the demanded position.

Syntax `SETCURSORPOS handle%, x%, y%`

Parameters	<i>handle%</i>	POINTER	I	handle of the window
	<i>x%</i>	INT(4)	O	position in x-axis
	<i>y%</i>	INT(4)	O	position in y-axis

Note

1. The *handle%* parameter could be:
 - ◆ The handle of the window (in this case the 0,0 is the left bottom corner of the NS-DK window),
 - ◆ the handle of a control (SELFFROMCONTROL%) (in this case the 0,0 is the left bottom corner of the control)
 - ◆ 0 and in this case the 0,0 is at the left bottom of the screen.

Example

```

*****
;*** Positionning the cursor at the bottom left of the screen
*****

SETCURSORPOS 0,0,0

*****
;*** Positionning the cursor at the bottom left of the NSDK
; window
*****
SETCURSORPOS 0,0,Self%

*****
;*** Positionning the cursor at the bottom left of the PB_TEST
;control of a NS-DK window
*****
SETCURSORPOS 0,0, SELFFROMCONTROL%(PB_TEST)

```

See also `GETCURSORPOS`

SetAttachedChildFlags instruction

Allow to position one or more flags to the attached child window.

Syntax **SetAttachedChildFlags** *wnd, kind%, flags%*

Parameters	<i>wnd</i>	POINTER	I	handle of the parent window
	<i>kind%</i>	INT(1)	I	position of the child window
	<i>flags%</i>	INT(1)	I	a selection of one or more ACF_ *% constants separated by a '+'

Notes

1. By attaching a window, the user creates a parent-child link between the two windows and defines the positioning of the child window in relation to the parent window (at the top, bottom, left or right of the parent window).
2. The positioning of the child window is defined by one of the CHILD_ *% constant.

Example

```
Local  Pointer WndAtt2

OpenH  WATTACH2, 0, WndAtt2
WATTACH2(WndAtt2).T1 = "Right"
SetBackCol COL_LightCyan% To WndAtt2
AttachChildWindow MainWindow%, CHILD_PostfixArea%, WndAtt2
SetAttachedChildFlags MainWindow%, CHILD_PostfixArea%, ACF_Resizable%
; Positioning the attached window at the right, it is resizable
```

See also GetAttachedChildFlags%, ACF_ *% constants

GetAttachedChildFlags% function

Allows to retrieve one or more flags corresponding to the ACF_%% constants.

Syntax **GetAttachedChildFlags%** *wnd*, *kind%*

Parameters	<i>wnd</i>	POINTER	I	handle of the parent windows
	<i>kind%</i>	INT(1)	I	positioning of the child window

Return value INT(1)

See also ACF_%% constants, SetAttachedChildFlags

ACF_*% constants

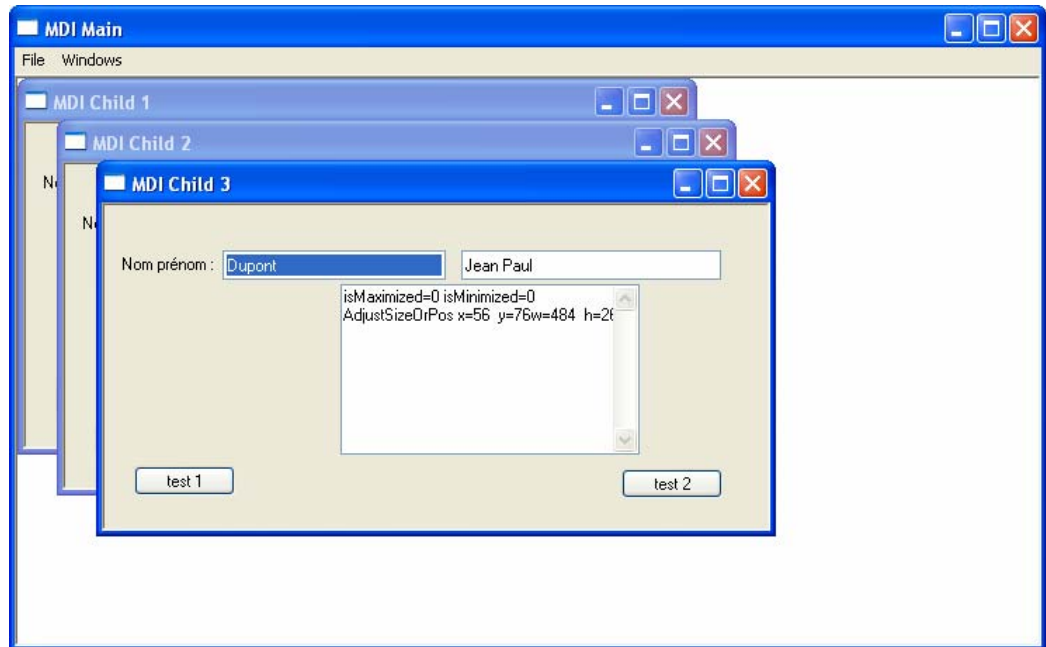
Values characterizing the attached child window with the SetAttachedChildFlags instruction.

Syntax **ACF_MDIDeactivate%**
 ACF_Resizeable%

Notes

1. The ACF_MDIDeactivate% constant makes it possible to deactivate the MDI child window menu, when the focus is positioned on the attached child window.
2. The ACF_Resizeable% constant makes it possible to resize the MDI child window.
3. These constants are used in the third parameter of the SetAttachedChildFlags instruction.
4. That is to say a MDI window having two attached windows WA and WB, whose WB is positioned with the ACF_MDIDeactivate% flag, and a MDI child window. The menu of the MDI child window does not appear automatically while selecting WA. It is necessary to click on the MDI child window to reveal the menu.
5. MDI windows make it possible to open simultaneously several MDI child windows. These child windows are attached since their creation to the parent window.

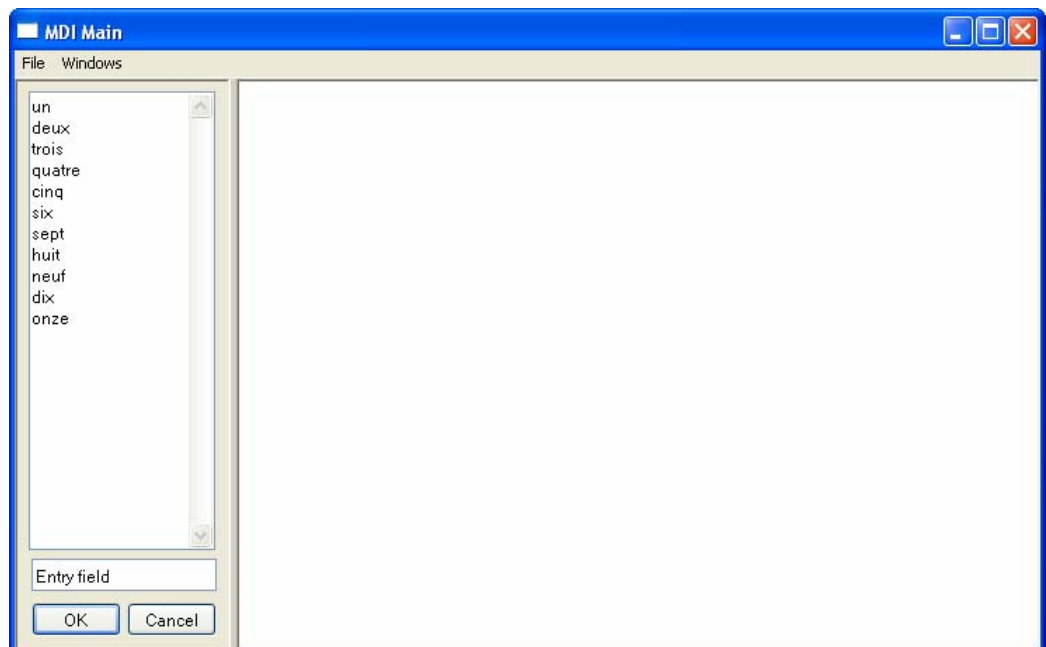
```
Open WMDICHLD, MainWindow%, Wnd
```



6. An attached window is a window of Dialog class opened without handle of the parent window and attached with ATTACHCHILDWINDOW instruction in order to be integrated into the parent window.

```
OpenH WATTACH, 0, WndAtt
```

```
AttachChildWindow Self%, CHILD_PrefixArea%, WndAtt
```



7. Their internal declaration is:

```
CONST ACF_MDIDeactivate% $01
CONST ACF_Resizeable%    $02
```

Example

```
Local   Pointer WndAtt2

OpenH WATTACH2, 0, WndAtt2
WATTACH2(WndAtt2).T1 = "Bottom"
SetBackCol COL_LightGreen% To WndAtt2
AttachChildWindow MainWindow%, CHILD_BotArea%, WndAtt2
SetAttachedChildFlags MainWindow%, CHILD_BotArea%, \
ACF_Resizeable%
```

See also

GetAttachedChildFlags%, SetAttachedChildFlags

NSMisc library

For recall, this miscellaneous library contains a large number of diverse functions and instructions, covering areas such as file handling, disk handling (directories, available space, file searches, etc.), random number generation, ASCII/EBCDIC conversion, and so on.

Managing traces

SetNSGenTrace	2-20
MyTrace	2-21
NS_TRACE	2-22
SEG_NSAGENTRACE	2-24
GTE_ *%	2-25

Managing the anchoring controls

NS_AS_*	2-26
---------------	------

SetNSGenTrace instruction

Allows you to modify the traces formatting by providing a pointer on a MyTrace function.:

Syntax **SetNSGenTrace** *traceFunc*

Parameter *traceFunc* POINTER I pointer towards a MyTrace function

See also myTrace

MyTrace instruction

Allows you to modify the traces formatting.:

Syntax **MyTrace** *trace, line%, info*

Parameters	<i>trace</i>	SEGMENT	I	reference on the SEG_NSGENTRACE segment
	<i>line%</i>	INT	I	line number of the source file
	<i>info</i>	POINTER	I	not used currently

Notes

1. The *line%* parameter corresponds to the line number where the event of trace occurred. The first line of the window event is 1.
2. The *info* pointer is reserved for a later use and could point on additional information. Currently, it is null for all traces.

See also SetNSGenTrace

NS_TRACE instruction

Allow to integrate the user traces in the NCL programs.:

Syntax **NS_TRACE** *message\$*

Parameter *message\$* CSTRING I user traces

Note

1. By default, the formatting of traces follows the next example:

```

NSDK:14:32:49
*****
NSDK:14:32:49 NSGENTRACE-<TRC> { ID in BINARY/FILE(LINE)
NSDK:14:32:49 NSGENTRACE-<TRC> } ID in BINARY/FILE(LINE) (CLOCK=CPU)
NSDK:14:32:49          \ \ \          \          \          \_____CPU
time between
NSDK:14:32:49          \ \ \          \          \          \
enter and leave
NSDK:14:32:49          \ \ \          \          \ _____
line in FILE
NSDK:14:32:49          \ \ \          \          \ _____
ressource FILE
NSDK:14:32:49          \ \ \          \_____BINARY
(dll or exe)
NSDK:14:32:49          \ \ \_____ID of
function/instruction/event
NSDK:14:32:49          \ \_____Enter or leave
funct/instr/event
NSDK:14:32:49          \_Generated trace reason 'instr', 'funct',
'event' for
NSDK:14:32:49          respectively instruction, function and
control event
NSDK:14:32:49
*****
NSDK:14:32:49 NSGENTRACE-event { SEQ.EXECUTED in TEST.EXE/TEST.SCR(1)
NSDK:14:32:49 NSGENTRACE-instr  { GO3 in TEST.EXE/SEQ.NCL(137)
NSDK:14:32:50 NSGENTRACE-instr  } GO3 in TEST.EXE/SEQ.NCL(139)
(CLOCK=0001.047)
NSDK:14:32:50 NSGENTRACE-event } SEQ.EXECUTED in TEST.EXE/TEST.SCR(2)
(CLOCK=0001.047)
NSDK:14:32:55 NSGENTRACE-event { <CLIENT>.TERMINATE in TEST.EXE/TEST.SCR(1)

```

```
NSDK:14:32:55 NSGENTRACE-event } <CLIENT>.TERMINATE in TEST.EXE/TEST.SCR(2)
(CLOCK=0000.000)
```

See also SEG_NSGENTRACE segment, GTE_ *% constants

SEG_NSAGENTTRACE segment

Syntax **SEGMENT SEG_NSAGENTTRACE**

version

trEvt

module

res

ctrl

id

descr

ENDSEGMENT

Fields	<i>version</i>	INTEGER	I	checking version of the segment
	<i>trEvt</i>	INTEGER	I	trace event (GTE_%% constants)
	<i>module</i>	CSTRING	O	EXE or DLL name
	<i>res</i>	CSTRING	O	NCL/SCR/TPL resource file name
	<i>ctrl</i>	CSTRING	O	for the NCL events only, "<CLIENT>" for the window
	<i>id</i>	CSTRING	O	identifier (function, instruction, event)
	<i>descr</i>	POINTER	I	description of the information pointer content

See also GTE_%% constants, NS_TRACE

GTE_%% Constants

Values for the GTE_%% trace events. .

Syntax

GTE_ENTERINSTR%
GTE_LEAVEINSTR%
GTE_ENTERFUNC%
GTE_LEAVEFUNC%
GTE_ENTEREVENT%
GTE_LEAVEEVENT%

Notes

1. Their meaning is the following:

GTE_ENTERINSTR%	beginning of an instruction
GTE_LEAVEINSTR%	end of an instruction
GTE_ENTERFUNC%	beginning of a function
GTE_LEAVEFUNC%	end of a function
GTE_ENTEREVENT%	beginning of a window event
GTE_LEAVEEVENT%	end of a window event

2. Their internal declaration is :

```
; GTE = Generator Trace Event
CONST GTE_ENTERINSTR% 1
CONST GTE_LEAVEINSTR% 2
CONST GTE_ENTERFUNC% 3
CONST GTE_LEAVEFUNC% 4
CONST GTE_ENTEREVENT% 5
CONST GTE_LEAVEEVENT% 6
```

See also SEG_NSGENTRACE, NS_TRACE, MyTrace, SetNSGenTrace

NS_AS_* Constants

Values for the traces events. To use with the dynamic parameter .ANCHOR.

Syntax

NS_AS_LEFT
NS_AS_RIGHT
NS_AS_TOP
NS_AS_BOTTOM
NS_AS_PLEFT
NS_AS_PRIGHT
NS_AS_PTOP
NS_AS_PBOTTOM
NS_AS_NONE
NS_AS_NSDK_DEFAULT
NS_AS_WINDOWS_DEFAULT
NS_AS_BOTTOM_RIGHT
NS_AS_TOP_RIGHT
NS_AS_WIDTH
NS_AS_WIDTH_HEIGHT

Notes

1. Use the dynamic parameter .ANCHOR with these constants.
2. Their meaning is :

NS_AS_LEFT	Fixation to the left
NS_AS_RIGHT	Fixation to the right
NS_AS_TOP	Fixation at the top
NS_AS_BOTTOM	Fixation at the bottom
NS_AS_PLEFT	Fixation proportionnally to the left
NS_AS_PRIGHT	Fixation proportionnally to the right
NS_AS_PTOP	Fixation proportionnally at the top

NS_AS_PBOTTOM	Fixation proportionnally at the bottom
NS_AS_NONE	No anchoring. The edges of the control are not attached to the edges of the window.
NS_AS_NSDK_DEFAULT	Fixation at the left bottom of the control.
NS_AS_WINDOWS_DEFAULT T	Fixation at the left top of the control.
NS_AS_BOTTOM_RIGHT	Fixation at the right bottom of the control.
NS_AS_TOP_RIGHT	Fixation at the right top of the control.
NS_AS_WIDTH	Fixation of the width.
NS_AS_WIDTH_HEIGHT	Fixation of the width and the height.

3. Their internal declaration is :

```

const NS_AS_LEFT      "L "
const NS_AS_RIGHT     "R "
const NS_AS_TOP       "T "
const NS_AS_BOTTOM    "B "
const NS_AS_PLEFT     "%L "
const NS_AS_PRIGHT    "%R "
const NS_AS_PTOP      "%T "
const NS_AS_PBOTTOM   "%B "

const NS_AS_NONE      " "
const NS_AS_NSDK_DEFAULT "B L "
const NS_AS_WINDOWS_DEFAULT "T L "
const NS_AS_BOTTOM_RIGHT "B R "
const NS_AS_TOP_RIGHT  "T R "
const NS_AS_WIDTH      "B L R "
const NS_AS_WIDTH_HEIGHT "T B L R "

```

Example

```

local control c
firstcontrol (self%,C)
while miscerror% = 0
    c.ANCHOR = NS_AS_PTOP & NS_AS_PBOTTOM & NS_AS_PLEFT &
NS_AS_PRIGHT
    nextcontrol (c)
EndWhile

```

NSSEQ library

NS-DK 5.00 provides a new sequence library NSSEQ allowing you to manage the client-server.



This library is specific to NS-DK. It corresponds to the NSTHFORM library for NatStar.



For more informations, please refer to the chapter 2 of the "Services Libraries Reference - Volume 4 - NS-DK APIs" manual.

Chapter 3

NCL language



Language NCL was improved to take into account technological innovations.

***You will find in
this chapter***

- The new verbs of the NCL language
- The modified verbs of the NCL language

Table of contents

Verbs	3-3
-------------	-----

Verbs

Some verbs have been added in this version 5.00, others have been modified. Here is the list, each verbs being is preceded by the mention ' A' (for addition) or ' M' (for modification). This chapter detailed the description of these verbs.

- A .ANCHOR dynamic parameter
 - M POS% function
 - M INSERT\$ function
 - M STRING operator
 - M NEW instruction
 - A Dynstr type
 - M POINTER type
 - M PARAM12% and PARAM34 parameters
 - M PASS, SEND, POST, (STR)CALL(H), (STR)OPEN(H), (STR)OPENS(H)
instructions
-

.ANCHOR dynamic parameter

The .ANCHOR dynamic parameter makes it possible to anchor the affected control to the window which contains it.

The anchoring of controls allows them, at the time of the window resizing, to adapt to the new size of the window and to take the suitable position.

For example, if a control Push Button is anchored to the bottom right edge of the window, it is positioned to be remain always at the same distance of the bottom right edge of the window.

If a control is not anchored, in case of window resizing, its position in relation to the edges of the window is modified.

Syntax **.ANCHOR**

Note

1. To position the anchoring type, use the NS_AS_* constants documented in the chapter 2 of this manual.

POS% function

Returns the position of a string in another string.

Syntax **POS%** (*search-string*, *source-string*, *order*, *start-at-position*)

Notes

1. Position = 1 corresponds to the index of the first character in the source string, regardless of the string's type (STRING or CSTRING).
2. The function returns 0 if the string is not found.
3. The two last parameters are optional. The *order* parameter can be used without *start-at-position*. The *start-at-position* parameter can not be used without the *order* parameter.
4. The *order* parameter indicates the rank of the searched occurrence.
5. The *order* and *start-at-position* parameters begin to 1

Example 1

```
MOVE POS% ("SYS", "NAT SYSTEMS") TO I% ; places 5 in I%
```

Example 2

```
local Dynstr lib_doc$  
lib_doc$ = "Declaration Declaration"  
message "POS ", POS%("Decla", lib_doc$, 1 ) ; returns 1  
message "POS ", POS%("Decla", lib_doc$, 1, 1 ) ; returns 1  
message "POS ", POS%("Decla", lib_doc$, 2 ) ; returns 13  
message "POS ", POS%("Decla", lib_doc$, 2, 1 ) ; returns 13  
message "POS ", POS%("Decla", lib_doc$, 1, 5 ) ; returns 13
```

See also COPY\$, DELETE\$, FILLER\$, INSERT\$

INSERT\$ Function

Inserts a string into another string at a specified position and returns the resulting string.

Syntax **INSERT\$** (*source-string*, *destination-string*, *insert-position*) [, *nb-char-dest-string-del*])

Notes

1. *position* = 1 corresponds to the first character of the destination string, regardless of the type of string (STRING or CSTRING).
2. The new optional parameter *nb-char-string-dest-del* allows you to define the number of characters to delete in the destination string during the insertion. This option allows you basically to replace a certain number of characters in a string.
3. Do not confuse with the INSERT instruction.

STRING Operator

Converts to a string (Pascal format).

Syntax **STRING** *integer* / *real number* / *string* / (*integer*, *base*)

Notes

1. STRING can also be used as a "type" with LOCAL, GLOBAL, SEGMENT, FUNCTION and INSTRUCTION declarations.
 2. In NCL, the STRING and CSTRING types can be used interchangeably, which explains why there is only one operator for converting to a character string.
 3. Two new parameters *integer* and *base* allow you to convert a decimal number into another base (2, 8 or 16).
-

NEW Instruction

Allocates a data area in memory and returns the address of the allocated memory area. A third syntax has been inserted in the newly versions of Nat System products.

Syntaxes

NEW *segment-name* / *size*, *mem-address* [, *public-name*]

NEW @*CastedPointer* [, *public-name*]

NEW [[*table-without-size*]] *segment-name* / *size*, *var-segment-address* / [[*table-without-size*]]@*ref-variable* [, *public-name*]

Notes

1. The first parameter determines how the data area will be structured internally.
 - ◆ If it specifies a segment name, the area will have the same structure as the segment.
 - ◆ If it specifies a numeric value, the area will not be structured and will be identified by its *size* expressed in bytes.
2. *public-name* is optional. This parameter specifies the variable's public name and is typically used when exchanging data between applications.
3. *mem-address* is set to null if NEW was unable to allocate the memory required.
4. The second syntax allows you to assign a segment of the same type as the type of pointer passed as a parameter and assigns its handle to a @*CastedPointer* pointer.
5. The third syntax allows you to allocate a table that the size had not been defined previously.
6. Use the DISPOSE instruction to free the memory allocated.
7. Remember that the address of a previously declared variable can be obtained directly by coding "@" followed by the variable name.

DYNSTR type

Dynamic character string type used within the LOCAL, SEGMENT, FUNCTION and INSTRUCTION instructions. The definition of the character string size as well as the type allocation are not necessary.

Type DYNSTR makes it possible to communicate more simply with COM/DCOM modules or Web Services.

Type DYNSTR behaves overall like CSTRING type. It will be converted automatically into CSTRING in case of allocation towards a variable of this type.

Notes

1. A DYNSTR string cannot be used with the instructions FILL and MOV. Indeed, these instructions require to know as a preliminary the length of the string.
2. A DYNSTR string can end or not by a \$ sign. However, DS variable does not correspond to variable DS\$.
3. As a whole, the dynamic strings are used like CSTRING. However, their size is not given at the time of the code writing and of the program compilation. An expression can mix arbitrarily the four types of strings, but certain behaviours are different (particularly, with the expressions like S\$(3..5)).
4. The dynamic variables do not make it possible to exchange more than 255 characters with controls.
5. Certain operators (& and &&) and some standards functions and instructions (COPY\$, DELETE\$, INSERT\$, SKIP, LSKIP, RSKIP, UPCASE, LOWCASE) of the NCL accept the native type Dynstr directly without no preliminary conversion into CSTRING.
6. With regard to the passage of parameter per address to one function or instruction, one can not only pass the variables of the DYNSTR type with this kind of parameter, but also (only for the variables simple, not for the tables) of the variables (local, total, fields of segments or parameters) declared like Var{litteral-integer-constant} CString}. In this case, the length maximum of the parameters contents is of 255 characters per default so not specified since the last variable is not a DYNSTR. Truncation during the modifications of the parameters contents (and thus of variable) is then automatic.

7. You must use imperatively the NEW instruction to allocate dynamically a segment containing one or more variable of DYNSTR type. The NEW instruction adds in the heading of the memory block returned a pointer allowing standard instruction NCL DISPOSE to release the DYNSTR of the segment or of segment table.
8. It is possible to use a variable of the DYNSTR type as a variable host in a SQL query (SQL_EXEC, SQL_EXECSTR, SQL_EXECCSTR). However, it is completely excluded to use variables of segment type containing one or more fields of DYNSTR type (even indirectly via one field of the segment type), since the segment is exchanged like a binary block with the database, which would cause all kinds of bug in the event of presence of DYNSTR.

Example

```
; Declaration of a global variable
Global DynStr ds, DynStr table_ds$[3][4]
; Declaration of a local variable
Local DynStr ds$, DynStr table_ds[3][4]
; Declaration of a function or instruction parameter
MaFonction%(DynStr ds, DynStr@ ref_table_ds[][10])
; Declaration of a returned function type
Function MaFonction2$(i%) Return DynStr
; Declaration of segment fields

Segment MonSeg
...
  DynStr ds$
...
  DynStr table_ds[3][4]
...
  DynStr table_sans_taille_ds2[][10]
EndSegment ; MonSeg
```

PARAM12% and PARAM34% parameters

The PARAM12% and PARAM34% functions were modified in the versions 5.00 to improve the portability 64 bits. The PARAM12 % and PARAM34 % functions are henceforth pointers. The PARAM1 %, PARAM2 %, PARAM3 % and PARAM4 % functions stay integer 16 bits.

From now on PARAM12% and PARAM34% are pointers. PARAM1%, PARAM2%, PARAM3% and PARAM4% stay 16 bits integer. Therefore, on a 64 bits platform, PARAM12% is no more equivalent to DWORD%(PARAM2%, PARAM1%) and PARAM34% is no more equivalent to DWORD

The fifteen following instructions have been impacted with these new functionalities : PASS, SEND, POST, CALL, CALLH, STRCALL, STRCALLH, OPEN, OPENH, OPENS, OPENSH, STROPEN, STROPENH, STROPENS, STROPENSH.

PASS, SEND, POST, (STR)CALL(H), (STR)OPEN(H), (STR)OPENS(H) instructions

These instructions have been impacted by the modifications of the PARAM12% and PARAM34% parameters which from now on are pointers.

If POINTER are passed in parameters, then PASS uses PARAM12% instead of PARAM1% and PARAM2% or PARAM34% instead of PARAM3% and PARAM4%.