NS-DK

Version 5 SP2 Edition 1

New features

Ref. n° `NSTR5SP2NEW00033`

Contents

Organization of this manual

NS-DK 5 SP2 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 two chapters.

Chapter 1 **General presentation**

This chapter introduces the new features of NS-DK 5 SP2.

Chapter 2 **New functions and instructions**

This chapter introduces the new functions and instructions of NS-DK 5 SP2.

Conventions

Typographic 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]	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 SP2 integrates new functionalities allowing you to optimize the applications developed with this tool. Nat System thus fulfills the technological and technical requirements of its users.

You will find in this chapter

- The new interface of NS-DK
- The new features in the NCL language.
- A new format of online help.

Table of contents

Vista support

The service pack 2 of NS-DK 5 extends the use of Windows platforms to the new platform Windows Vista 32-bit.

CHM Online Help

Nat System provides an online help in CHM format (Compiled HTML Help Format) that is compatible with Vista.

The CHM format being compiled HTML, the files occupy much less memory than the old help files (.HLP and .CNT).

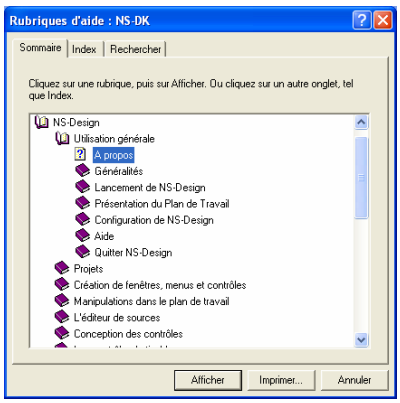
The CHM online help ***MUST IMPERATIVELY*** be located on the client station of each user.

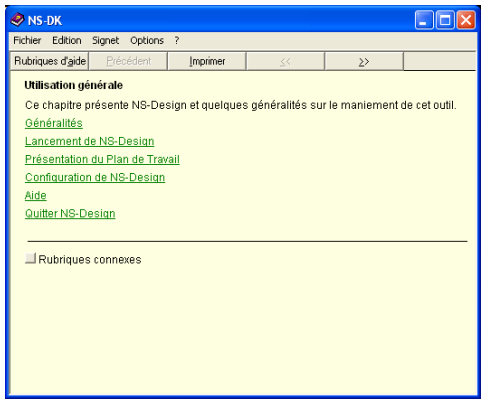
UNIX

The .CHM files are only compatible with Internet Explorer and ActiveX technologies. They are not accessible on Netscape Navigator or non-Windows platforms (Macintosh, UNIX,…) because the latter do not support ActiveX.

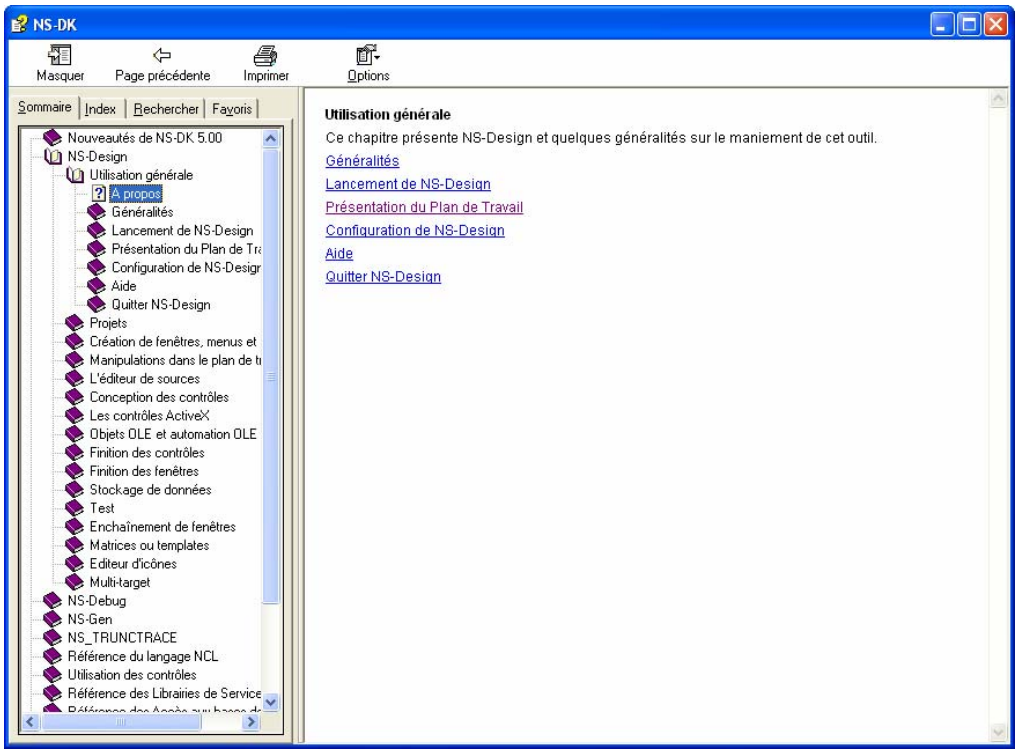
Single window

In the old on line help, the contents, the index and the fields of research were disposed in a separate window of the help headings.





The various components of the online help from now on are displayed in a single window.

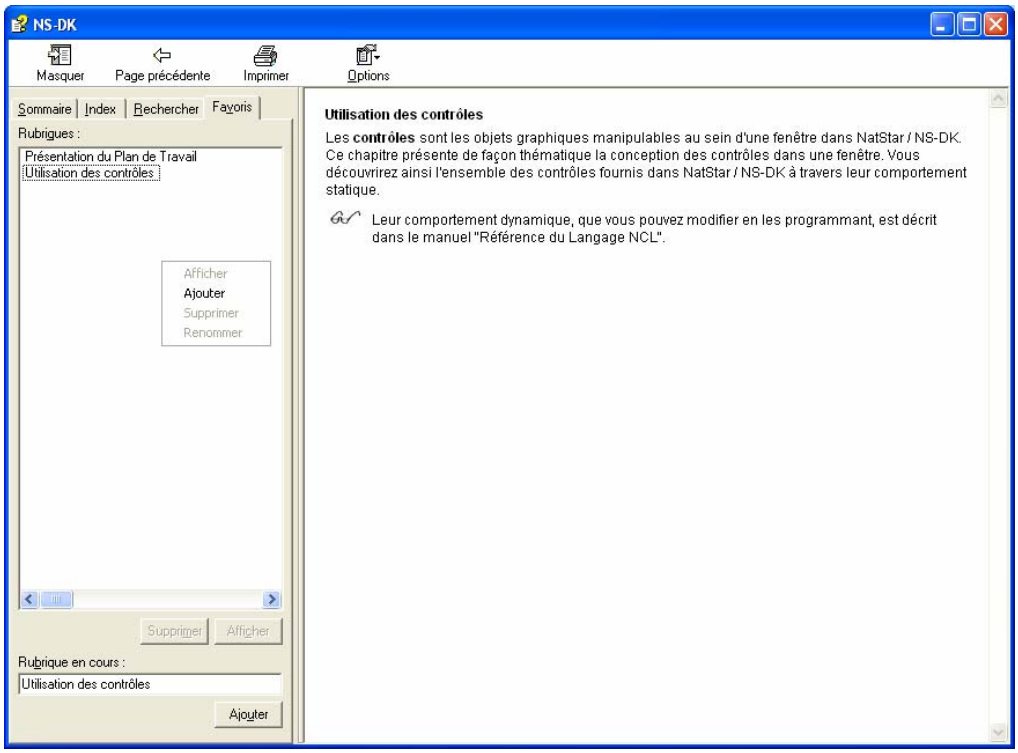


You can hide the left panel by activating the button 

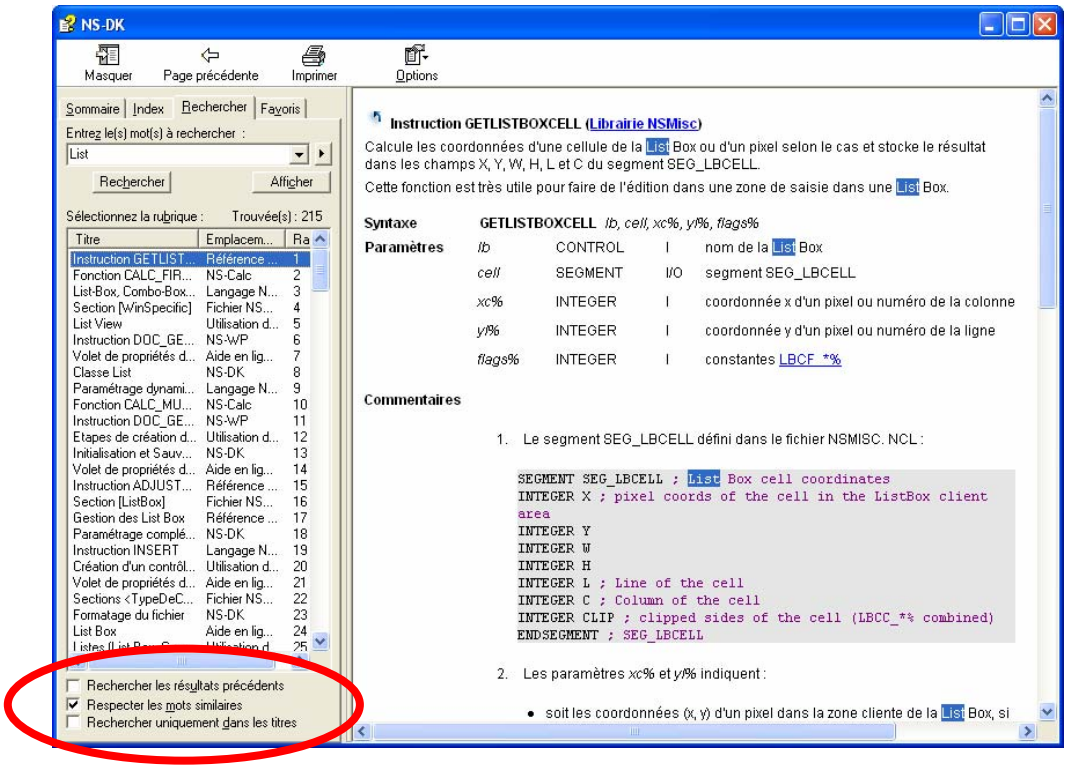
and displays it with the button 

.

A new *Favoris* tab (that is to say *Bookmarks* tab) makes it possible to refer help topics in order to check them up later.



The *Rechercher* tab (that is to say *Search* tab) improves the search criterias by proposing new functions or making them more visible than before: *Rechercher uniquement dans les titres (Search only in titles), Rechercher les résultats précédents* (*Search the following results*) and *Respecter les mots similaires (Respect the similar words).*



Inserting a CHM online help in NS-DK applications

To insert in your NS-DK applications an online help with CHM format, use the **HLPINITIALIZE** instruction with the new constant **HLPMODE_CHM%**.

To customize the call mode of the help, a new instruction **HLPHTMLHELPTYPKEY** and some new constants NS_HH_* (NSHelp library) have been created.

For more information, please refer to the chapter 4 of the "Services Libraries Reference – Volume 1 – GUI and Printing APIs" manual or the chapter 2 of this manual for an overview.

New NS-DK interface

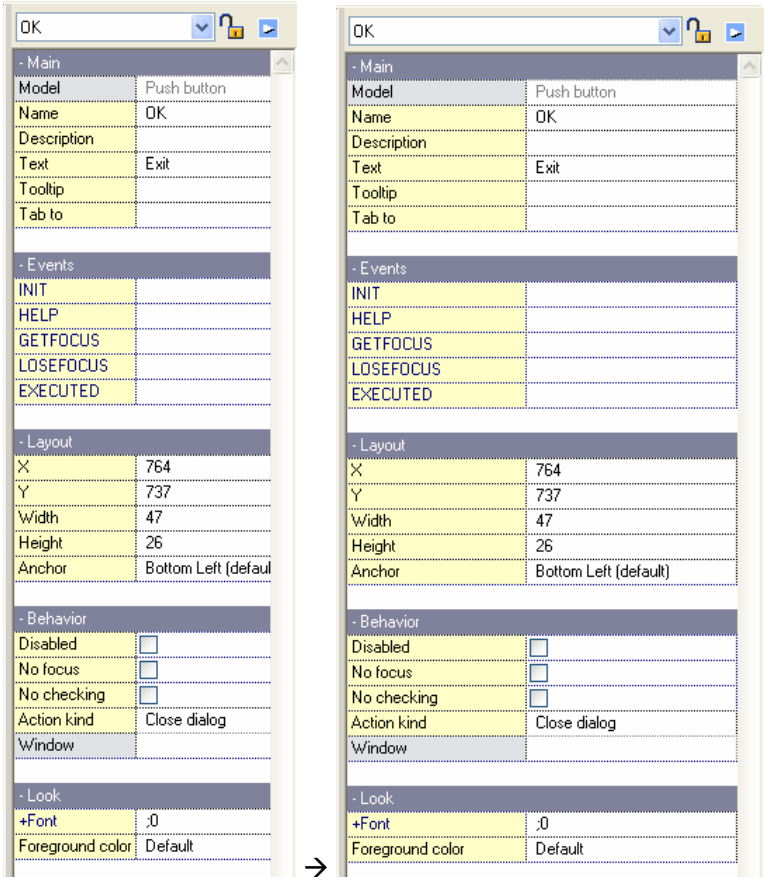
The interface of NS-DK evolves to improve the use of the tool.

.XNP project

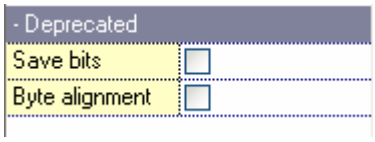
From now on, when you open a project, the type of project suggested by default is .XNP.

Properties pane

The **properties pane** of graphic controls is located on the right of the workspace. It is from now on resizable (with the mouse) for better visualize the various characteristics of the selected element.



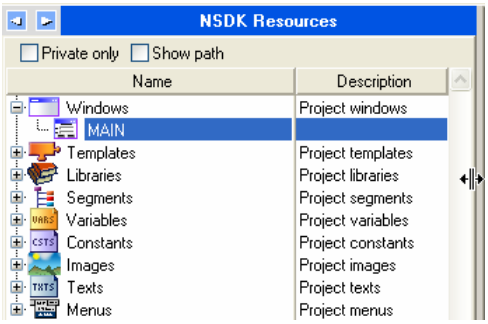
In addition, two properties of the windows are from now on obsolete *Save bits* and *Byte alignment*.



The *Save Bits* property was used to accelerate the windows display when the graphics cards were particularly slow. In addition, the *Byte alignment* property was useful mainly on the old OS/2graphics cards. These two properties are thus deprecated.

Browser resources

The browser resources is also from now on resizable.



Browsing in the NCL editor

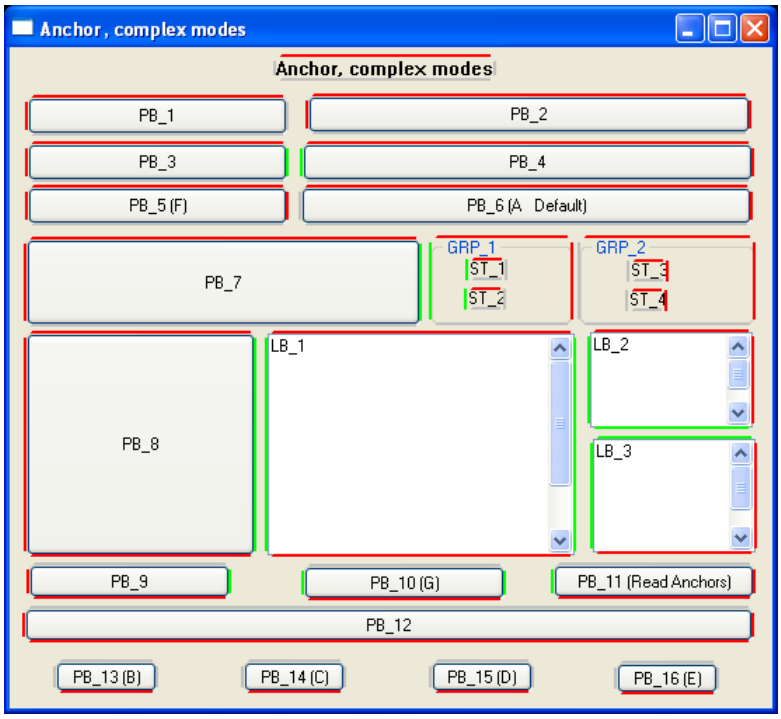
The arrowed keys [PageUp] and [PageDown] associated to the [Ctrl] key make it possible to browse more easily in the blocks of instruction (INSTRUCTION… ENDINSTRUCTION, IF… ENDIF,…) of NCL code.

Displaying the controls anchoring

The controls' anchors can be displayed using the new  *Display Anchors* icon seen in the toolbar (or through the *Set/Display Anchors* menu).

The anchors are displayed using the following colours:

- red; if that side is fixed onto a border,
- green; if that side is proportionally anchored onto the borders,
- grey; if that side has no anchor.



For more information, please refer to the chapter 21 of the "User Manual".

Bitmap transparency

There are new properties for the *Bitmap* controls.

- Transparency	
BMP Transparenc	Fixed Color
Corner pixel	Default
Transp Color	
Subst. Color	Default

BMP Transparency

Opaque

Corner Pixel

Fixed Color

Corner pixel

When the option **Corner pixel** is selected in the **BMP Transparency** field, you can choose which corner will be the base of the transparency colour.

Top Left

Bottom Left

Top Right

Bottom Right

Default : use the default color defined for the project (in general, the top left pixel).

Opaque

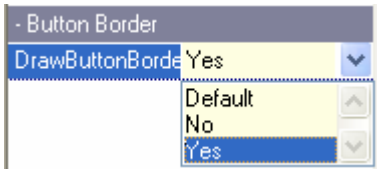
Transp Color

If you have choose **Fixed Color** in **BMP Transparency**, select in the **Transp color** field, the color that will be used in the transparency.

Subst Color

Color of transparency: color which will replace the transparent color. By default, the background color of the window (or button in the case where the button is drawn)

In addition, if the bitmap is a Push-Button type one, the **DrawButtonBorder** property allows you to redraw the bitmap borders.



Default: default value of the redrawing application /or not of the bitmaps type buttons.

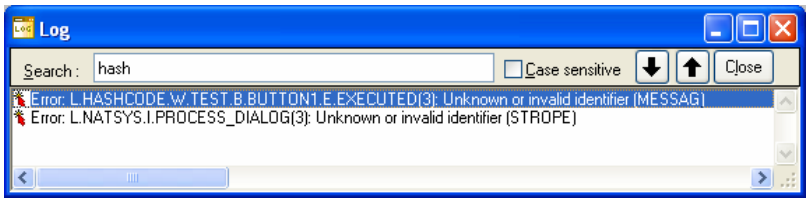
No: no drawing of the button border

Yes: drawing of the button border provided the following conditions are observed: In this mode if the fields Bitmap Released, Bitmap Pressed, Bitmap Disabled are not all informed or are identical the drawing of border is carried out, otherwise nothing is done.

Log window

By pressing the [Ctrl]+F key or the popup menu Find, a search pane appears at the top of the window. Enter in the entry field the string to search.

By ticking *Case sensitive*, the search will respect the case of the string entered. The two arrows allow you to search towards the top or the bottom.



Press the [Enter] key to begin the search. The *Close* button or the [Esc] key closes the search pane.

XML document security

Nat System proposes a new library named NWXSEC. It allows you to make your documents XML safe by using canonicalization.

For more information, please refer to chapter 8 of the manual "Services Libraries Reference - Volume 1 – GUI and Printing APIs".

User functions call

In order to personalise controls on client projects, it is now possible to define functions that will be called by NS-Design during development periods.

These control functions will be called:

- After confirming the NCL code ([F8] key) that belongs to a library or an event.
- Before and/or after launching a construction of the application (*Build*).
- From the NCL code editor (two functions).
- For any context (NCL editing, scr editing, compilation, etc.) (two **global** user functions).

For more information, please refer to the chapter 3 of the manual "Services Libraries Reference - Volume 4 - NS-DK APIs".

String truncature

Nat System presents NS-TruncTrace that offers a set of features that make it possible to investigate problems linked to string truncations when there is an exception of a program generated with NS-DK or NatStar.

It works with two investigation modes:

- Non-recompilation mode: When a string truncation has been detected, a warning window is displayed and/or an information message is generated in the trace file specified by the NS-TRACE environment variable.

- Recompilation mode: When a string truncation has been detected, an information message specifying the location of the truncation (file/row number) is generated in the file.

For more information, please refer to "NS-TruncTrace User Manual".

Improvement of the NCL language

Menu management

A new instruction **MNU_SET_STATUSTEXT** (NSCUST library) allows you to modify the status bar text.

A new constant **MENU_SEPARATOR_TOOLBAR%** allows you to specify a separator in the toolbar. Each separator must have in its *Modify menu item* box, the *Bitmap For* field specified by *Toolbar* or *Both*.

For more information, please refer to the chapter 8 of the manual "Services Libraries Reference - Volume 1 – GUI and Printing APIs".

Custom Control IE component

To make the management of the Custom Control IE easier, Nat System introduces dynamical parameters.

For more information, please refer to the chapter 9 of the manual "Services Libraries Reference - Volume 1 – GUI and Printing APIs".

NSLIB.INI file

Reading of the NSLIB.INI file

The reading of the NSLIB.INI file evolves to allows you to display the localization of the file read by the executable (in a message box), as well as the sections and parameters used (in the traces file).

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.

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
*****************************************************************************
```

[Winspecific] section

In the [Winspecific] section, the UseLookXP parameter evolves and a new parameter UseClipChildren is inserted.

`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.

For more information, please refer to "NSLIB.INI File" manual.

Tooltip

From now on, you can display a help bubble on several lines by entering the characters "#13" into the help bubble's text.

Example :

```
OK.tooltip = "First line "&#13&" – Second line"&#13&" – Third line"&#13&" –
Fourth line"
```

When the help bubble is displayed on a single line, the text will remain centred. If the help bubble is on several lines, the text will be left-aligned.

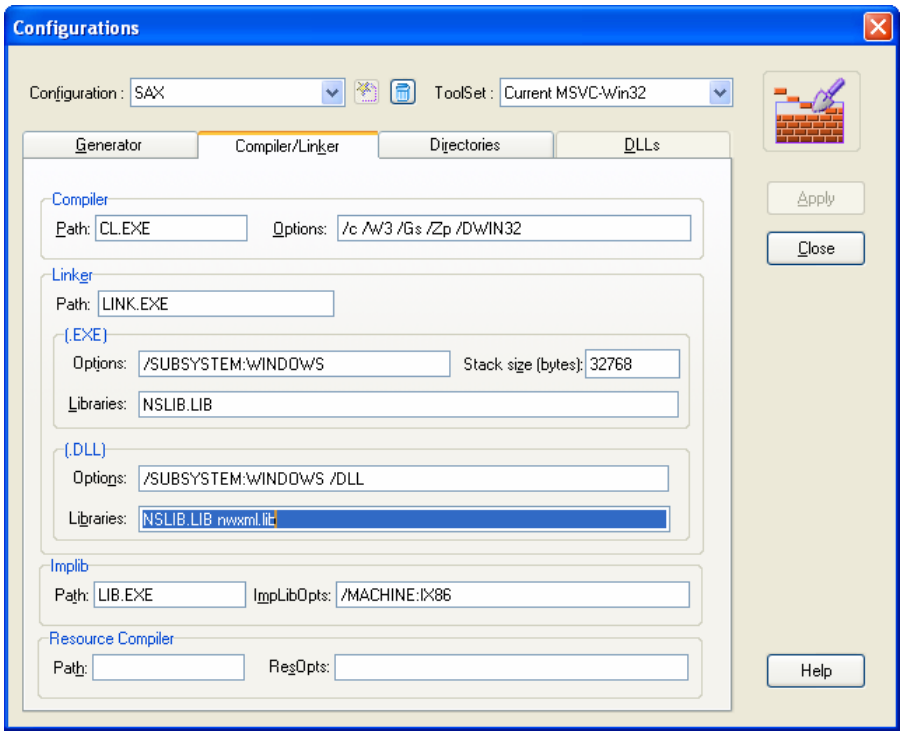
Compilation

In the configuration box (*Options\Build Configurations* … menu), you can from now on define a file name in the *Libraries* fields.

Example :

```
@(envVar)responsefile.rsp
```

This file contains the list of import libraries. If the environment variable (envVar parameter) is not defined, the file is searched in the root directory of the project (PROJDIR).



HTTP requests

To facilitate the use of the HTTP requests, Nat System introduces new functions and instructions to handle date of DynStr type:

- NS_HTTP_GET_HEADEREX
- NS_HTTP_GETLINEEX
- NS_HTTP_POSTEX
- NS_HTTP_READEX

For more information, please refer to the chapter 5 of the manual "Services Libraries Reference - Volume 1 – GUI and Printing APIs", or the chapter 2 of this manual for an overview.

NSMisc library

The NSMisc library introduces new APIs for the DynStr type :

- Management of the text files exceeding 255 characters (T_READLNEX$, T_WRITELNEX).
- The F_LoadFile$ function that returns the whole content of a file in one and only string of DynStr type. And the F_SaveFile instruction that save text in a file.
- The GETLONGENV$ function that returns the value of a environment variable in a DynStr type string.

For more information, please refer to the chapter 4 of the manual "Services Libraries Reference - Volume 2 – Kernel APIs", or the chapter 2 of this manual for an overview.

NWXML library

The NWXML library introduces new functions allowing you to manage string of DynStr type: NWX_NODE_OUT_DYNSTR and NWX_DOCUMENT_OUT_DYNSTR.

For more information, please refer to the chapter 6 of the manual "Services Libraries Reference - Volume 3 – Communication APIs".

NS-PREX component

Nat System is offering a new component for communication between different tools, which is used to call a NS-DK application (as a DLL or executable) from/within a web browser, while sending it parameters.

This component works on all available browsers (Internet Explorer, Firefox, etc.) without ActiveX and without altering the security settings of the client system.

This component is used to load an NS-DK DLL and to call several instructions while sending it parameters. Otherwise, it is also used for running a NS-DK executable, while sending it parameters.

This component is very easy to use. A single line of JavaScript makes it possible to run the NS-PREX component call. Also, the description of the executables and DLLs that are to be run are found in a single .INI file, so that the JavaScript can be simplified.

For more information, please refer to "NS-PREX component" manual.

Chapter 2 New functions and instructions

This chapter presents the new functions and instructions introduced in NS-DK 5 SP2 for development optimization.

You will find in this chapter

- The new functions and instructions of the NSHELP, NSMISC, NSCUST, NSHTTP and NWXML libraries.

Contents

NSHelp Library

The Nat System help library provides a range of functions and instructions that enable a NS-DK application to include and use on-line help generated in the form of .HLP or .CHM files.

Initialize the online help

NS-DK 5 SP2 introduces a new constant HLPMODE_CHM% allowing you to load a CHM online help.

Call mode of the online help

NS-DK 5 SP2 introduces an instruction and its associated constants to manage the help opening.

HLPMODE_*% Constants

These constants are used by the HLPINITIALIZE instruction. They specify the help system to be used.

Syntax

HLPMODE_NATSYS%
HLPMODE_WINHELP%
HKLPMODE_CHM%

Notes

1. The meaning of these constants is as follows:
   - HLPMODE_NATSYS%
     specifies that the Nat System help system is to be used.
   - HLPMODE_WINHELP%
     specifies that the Windows help facility is to be used.
   - HLPMODE_CHM%
     specifies that the CHM format is to be used.
2. They are declared internally as follows:

```
CONST HLPMODE_WINHELP%         0
CONST HLPMODE_NATSYS%          4
CONST HLPMODE_CHM%             5
```

Example

```
HLPINITIALIZE HLPMODE_NATSYS%, " ", \
              "C:\NS-DK\HLP\NSHELP.HLP", " "
```

See also

HLPINITIALIZE

HLPHTMLHELPTYPKEY instruction

Allows you to specify the call mode of the help.

Syntax	**HLPHTMLHELPTYPKEY** *NS_HH_COMMAND%*			
Parameter	*NS_HH_COMMAND%*	INT(2)	I	NS_HH_* constant
See also	NS_HH_* constants, HLPMODE*%			

NS_HH_* constants

Used by HLPHTMLHELPTYPKEY instruction to specify the call mode of the help.
.

Syntax

NS_HH_DISPLAY_TOPIC

NS_HH_DISPLAY_TOC

NS_HH_DISPLAY_INDEX

NS_HH_KEYWORD_LOOKUP

NS_HH_ALINK_LOOKUP

NS_HH_HELP_CONTEXT

Notes

1. These constants are used in the HLPHTMLHELPTYPKEY instruction. They have the following meaning :
   - NS_HH_DISPLAY_TOPIC

     Call of the help from a topic.htm.
   - NS_HH_DISPLAY_TOC

     Call of the help from a topic.htm and selection of the Summary tab.
   - NS_HH_DISPLAY_INDEX

     Call of the help and selection of the Search tab and highlight the search field.
   - NS_HH_KEYWORD_LOOKUP

     Call of the help and opening the identified page by the index passed in parameter.
   - NS_HH_ALINK_LOOKUP

     Call of the help and opening the identified page by the secondary index passed in parameter. A secondary index identifies a topic htm but does not appear in the list of the indexes on the Search tab.
   - NS_HH_HELP_CONTEXT

Call of the help and opening the identified page by ID of MAP passed in parameter. This ID is created during the generation of the help and is visible on the project allowing you to generate the help.

2. Their internal declaration is :

```
CONST NS_HH_DISPLAY_TOPIC                          0
CONST NS_HH_DISPLAY_TOC                            1
CONST NS_HH_DISPLAY_INDEX                          2
CONST NS_HH_KEYWORD_LOOKUP                         3
CONST NS_HH_ALINK_LOOKUP                           4
CONST NS_HH_HELP_CONTEXT                           5
```

Examples

In the INIT event of the window, we have:

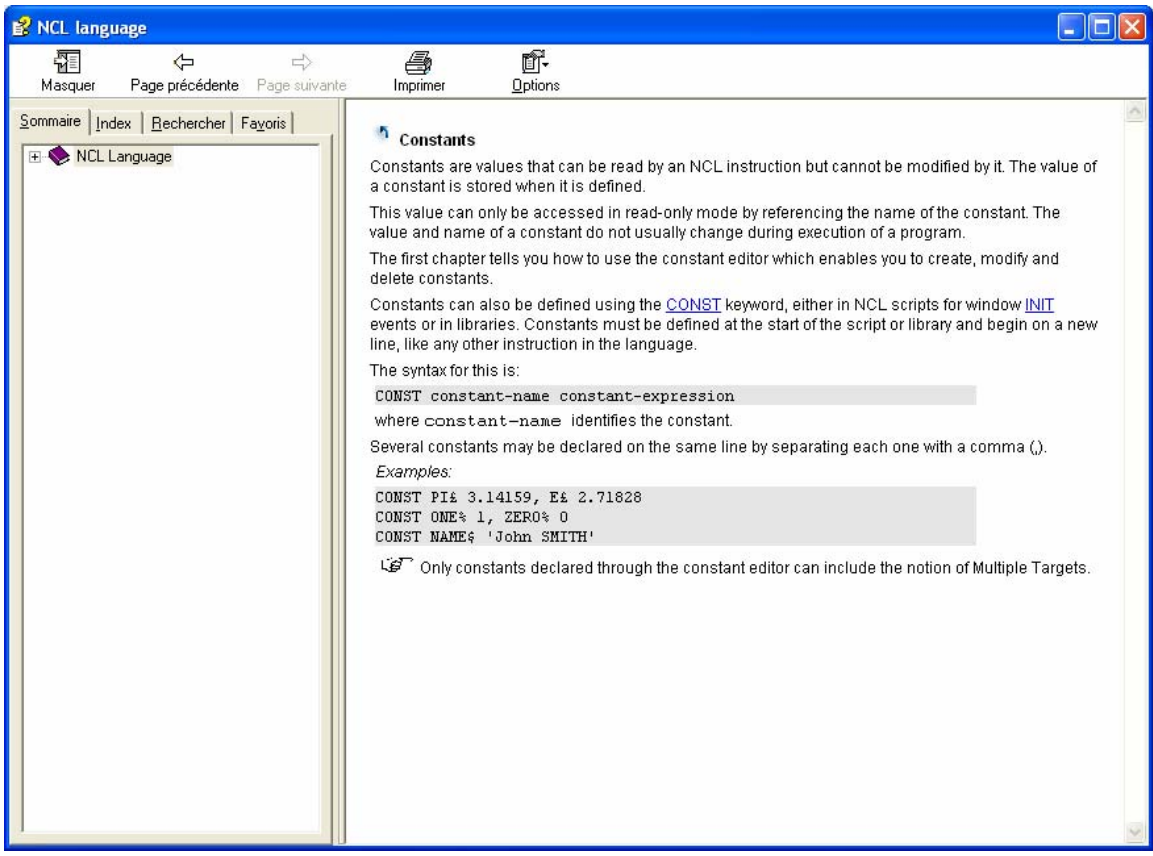
```
HLPINITIALIZE HLPMODE_CHM%, MAINWINDOW%,"",NSADE.CHM", ""
```

On the EXECUTED event of the help button, we have the following call choices:

- *Call with NS_HH_DISPLAY_TOPIC :*

```
HLPHTMLHELPTYPKEY NS_HH_DISPLAY_TOPIC
HLPOPEN MAINWINDOW%,"Constants.htm"
```

Opening the help at the topic Constants.htm



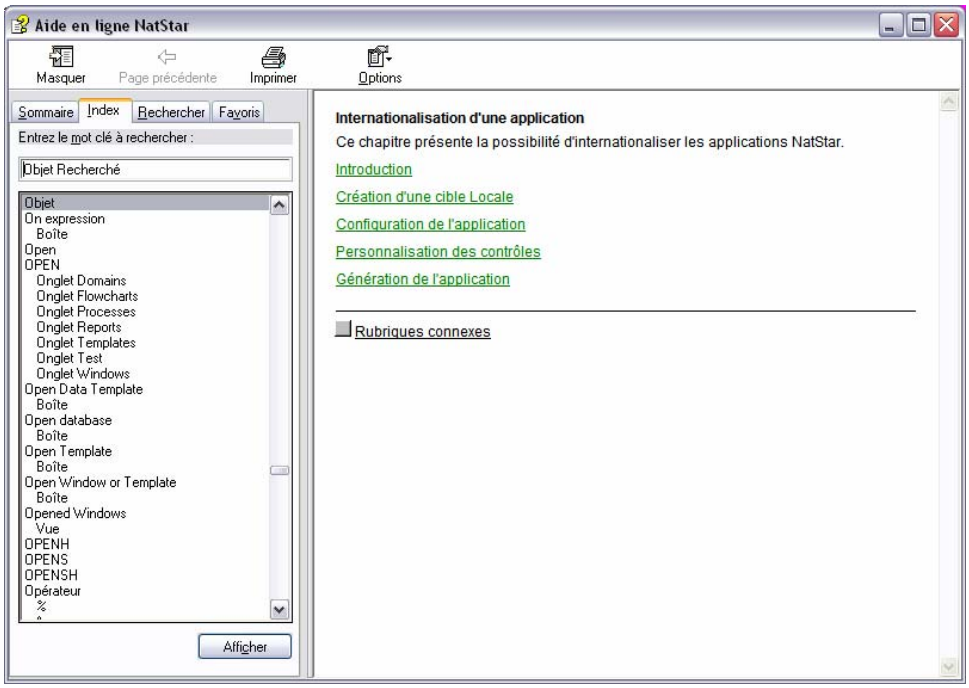
- *Call with NS_HH_DISPLAY_TOC :*

This call is the same as the following except that the Summary tab is selected if it wasn't during the last opening of the help.

- *Call with NS_HH_DISPLAY_INDEX :*

```
HLPHTMLHELPTYPKEY NS_HH_DISPLAY_INDEX
HLPOPEN MAINWINDOW%,"Objet recherché"
```

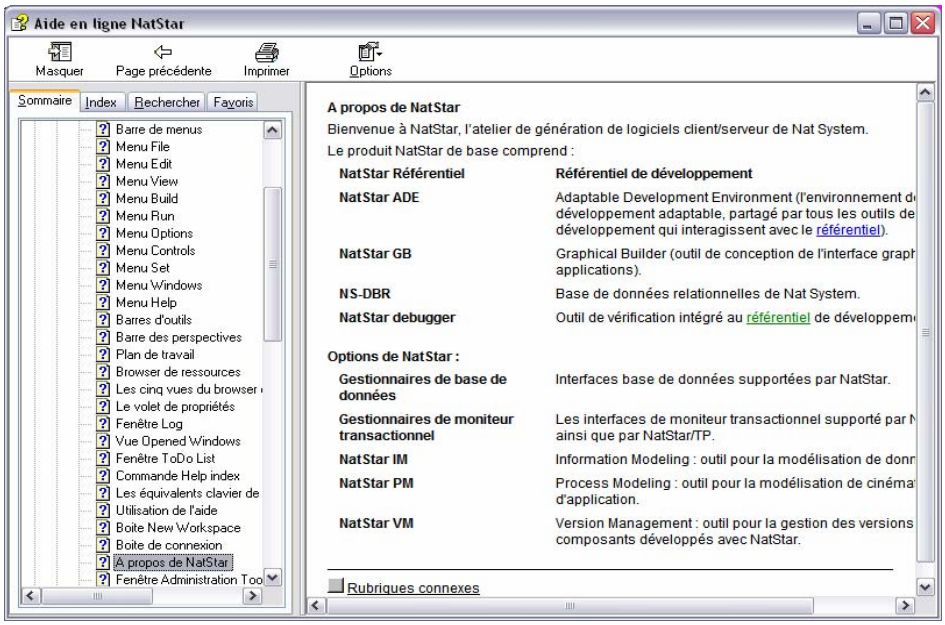
Opening the help with the Search tab selected and the keyword field is filled up.



- *Call with NS_HH_KEYWORD_LOOKUP*

```
HLPHTMLHELPTYPKEY NS_HH_KEYWORD_LOOKUP
HLPOPEN MAINWINDOW%,"About"
```

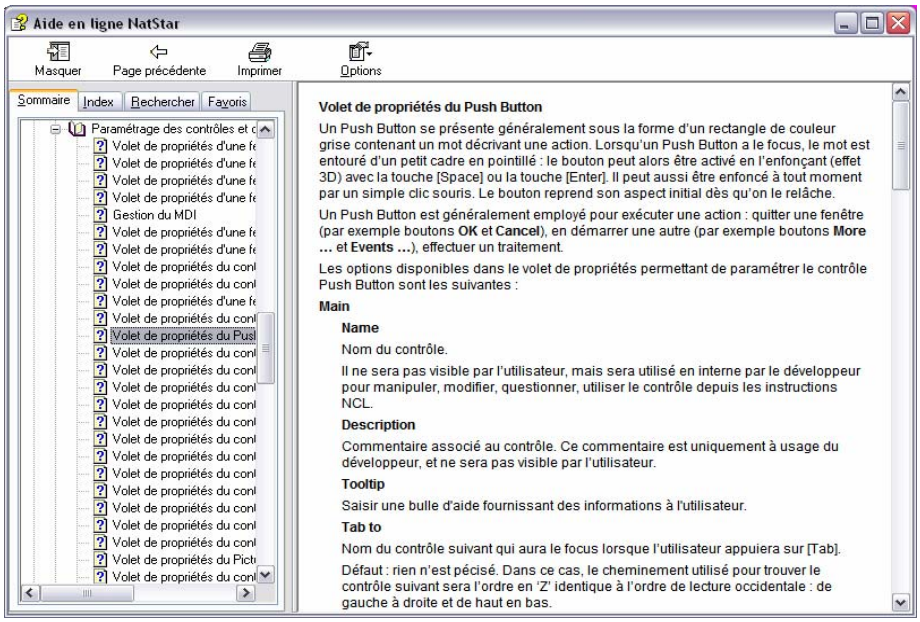
Opening the help with the topic referenced by the index 'About'.



- *Call with NS_HH_ALINK_LOOKUP :*

```
HLPHTMLHELPTYPKEY NS_HH_ALINK_LOOKUP
HLPOPEN MAINWINDOW%,"PUSH"
```

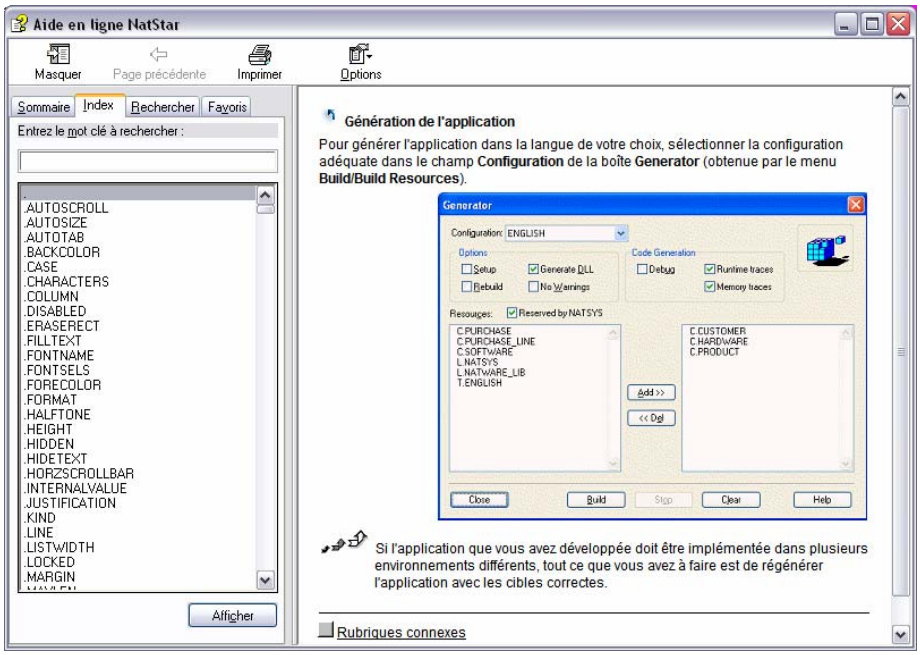
Opening the help with the topic referenced by the secondary index 'PUSH'.



- *Call with NS_HH_HELP_CONTEXT :*

```
HLPHTMLHELPTYPKEY NS_HH_HELP_CONTEXT
HLPOPEN MAINWINDOW%,"506"
```

Opening the help at the topic referenced by the ID equal to 506.



See also HLPHTMLHELPTYPKEY, HLPMODE, HLPINITIALIZE

NSMisc Library

These two new functions allow you to manage text file by using DynStr.

T_READLNEX$ Function

Returns the contents of the current line in an ASCII text file and positions the read pointer at the beginning of the next line.

Syntax **T_READLNEX$** (*file-handle*)

Parameter

file-handle	INT(4)	I	file handle

Return value DYNSTR

Notes

1. The only difference between T_READLN$ and T_READLNEX$ is that T_READLNEX$ allows you to return a string containing more than 255 characters.
2. See notes for T_READ$.
3. We recommend that you always call T_ERROR% after calling a T_*% function or instruction to check that it was executed correctly.

See also T_READLN$, T_READ%, T_READ$, T_READ#, T_READLN%, T_READLN#

T_WRITELNEX Instruction

Writes a specified expression to an ASCII text file and includes a line feed character at the end of the expression.

Syntax **T_WRITELNEX** *file-handle, expression*

Parameters

file-handle	INT(4)	I	file handle
expression	DYNSTR	I	expression to write

Notes

1. The only difference between T_WRITELN and T_WRITELNEX is that T_WRITELNEX allows you to write an expression of more than 255 characters.
2. The expression is automatically converted to a character string.
3. After writing to the file, the pointer will be positioned at the beginning of the next line.
4. T_WRITELNEX returns an error if the file was opened in read-only mode by T_OPEN%.
5. We recommend that you always call T_ERROR% after calling a T_*% function or instruction to check that it was executed correctly.

Example

```
; Let's assume that the file has been opened and that its
; handle has been copied into the H% variable
T_WRITELNEX H%, " Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello Hello "
T_WRITELNEX H%, " Good-bye "
; The last line in the file contains the string " Good-bye
; and the previous line contains the string " Hello Hello Hello Hello
;Hello … "
T_CLOSE H%
```

See also T_WRITELN, T_WRITE, T_WRITEEX

F_LoadFile$ Function

Return the totality of a content file, indicated in parameter, in only one character string of the DynStr type. Thus, the jumps of line are present like control characters.

Syntax **F_LoadFile$** *FileName$*

Parameter	*Filename$*	CSTRING	I	name of a file

Return value DYNSTR

Note

1. We recommend that you call MISCERROR% function after calling the F_LoadFile$ function to check that it was executed correctly.

Example

```
F_LoadFile$ "File02.txt"
IF MISCERROR% = 0
   MESSAGE "OK", "Load of the file is completed!"
ELSE
   MESSAGE "Error" && MISCERROR%, "Load of the file is not completed"
ENDIF
```

See also F_SaveFile

F_SaveFile Instruction

Allows you to save text in a file.

Syntax **F_SaveFile** *FileName$, Text$*

Parameters	*FileName$*	CSTRING	I	name of a file
	Text$	DYNSTR	I	content of a file

Return value DYNSTR

Note

1. We recommend that you call MISCERROR% function after calling the F_SaveFile instruction to check that it was executed correctly.

Example

```
F_SaveFile "Fichier1.txt", "The F_SaveFile instruction allows you to \ save text in a file"
IF MISCERROR% = 0
   MESSAGE "OK", "Save is completed!"
ELSE
   MESSAGE "Error" && MISCERROR%, "Save is not completed"
ENDIF
```

See also F_LoadFile$

GETLONGENV$ Function

Returns the value of an environment parameter.

Syntax **GETENV$** (*env-string*)

Parameter

env-string	CSTRING	I	environment parameter

Return value DYNSTR

Note

1. This function returns character strings whatever its size. It is thus often more practical to use this function instead of GETENV$.

Example

```
MESSAGE " PATH variable ", GETLONGENV$(" PATH ")
MESSAGE " NS-INI variable ", GETLONGENV$(" NS-INI ")
```

See also GETENV$, ENVCOUNT%, ENVSTR$, PUTENV

NSCUST Library

The NSCUST library allows you to handle and manage the menus.

MNU_SET_STATUSTEXT instruction

Allows you to modify the status bar text.

Syntax **MNU_SET_STATUSTEXT** *hWin, statustext$*

Parameters

hWin	POINTER	I	window handle (parent window or child window in case of merge with MNU_MERGE% function).
nItem	CSTRING	I	character string to display in the status bar.

Note

1. This instruction works only with a window containing a menu **and** a status bar. That is to say when the user creates a menu with the MNU_OPEN instruction and the MENU_BOTTOMBAR% option.

See also MNU_OPEN, MENU_*% constants, MNU_MERGE%

MENU_*% Constants

Display options for a menu.

Syntax

MENU_SIMPLE_TOOLBAR%
MENU_TEXT_TOOLBAR%
MENU_BUBBLE_HELP%
MENU_BOTTOMBAR%
MENU_SEPARATOR_TOOLBAR%
MENU_SIMPLE_STATUSBAR% (obsolete)
MENU_TEXT_STATUSBAR% (obsolete)

Notes

1. These constants have the following meaning:

   `MENU_SIMPLE_TOOLBAR%`

   The menu is displayed with its toolbar. It replaces MENU_SIMPLE_STATUSBAR%.

   `MENU_TEXT_TOOLBAR%`

   The toolbar buttons display the menu item's text. It replaces MENU_TEXT_STATUSBAR%.

   `MENU_SIMPLE_STATUSBAR%`

   The menu is displayed with its toolbar. It's obsolete and replaced by MENU_SIMPLE_TOOLBAR%.

   `MENU_TEXT_STATUSBAR%`

   The toolbar buttons display the menu item's text. It's obsolete and replaced by MENU_TEXT_TOOLBAR%.

   `MENU_BUBBLE_HELP%`

   Help bubbles pop from the toolbar buttons.

   `MENU_BOTTOMBAR%`

   A status bar is displayed at the bottom of the screen. It displays the selected menu item's description (which comes from the *Description* field in the *Menu XXX* box).

   `MENU_SEPARATOR_TOOLBAR%`

   A separator in the toolbar. Each separator must have in its *Modify menu item* box, the *Bitmap For* field specified by *Toolbar* or *Both*.

2. They are declared internally as follows:

```
CONST MENU_SIMPLE_TOOLBAR% 1
CONST MENU_TEXT_TOOLBAR% 2
CONST MENU_BUBBLE_HELP% 4
CONST MENU_BOTTOMBAR% 8
CONST MENU_SEPARATOR_TOOLBAR% 16
```

NSHTTP Library

The NSHTTP library allows you to manage the HTTP requests.

NS_HTTP_GET_HEADEREX function

Retrieve the Header of a http response and assigned it to a buffer.

Syntax **NS_HTTP_GET_HEADEREX** *pHttp*

Parameters

pHttp	POINTER	I	connection's handle

Return value DynStr

Notes

1. This instruction is useful when processing a single header element is not needed.
2. It would be better to use the TEXT dynamic attribute to set the contents of an MLE from a DynStr.

Example 1

```
LOCAL POINTER http%
LOCAL DynStr ds

; S1 and S2 are two MLEs
S1 = ""
S2 = ""

; create HTTP connection object
http% = NS_HTTP_NEW
; connect to the given site
NS_HTTP_CONNECT http%, URL
NS_HTTP_GET http%

IF (NS_HTTP_ERROR <> 0)
   ERROR = NS_HTTP_GET_ERRORMSG
ENDIF

; get all header fields and add them to the text area
S1.TEXT = NS_HTTP_GET_HEADEREX(http%)

; read the content of the http response and write it to the text area
ds = NS_HTTP_READEX(http%)
; by using the .text attribute we could transfer a string greater than 255
; characters to an MLE
S2.text = ds

IF (NS_HTTP_ERROR <> 0)
   ERROR = NS_HTTP_GET_ERRORMSG
ENDIF
; free the http connection object
NS_HTTP_DISPOSE(http%)
```

Example 2

```
LOCAL POINTER http%
LOCAL DYNSTR buf$, DYNSTR line$

S1 = ""
S2 = ""

; get HTTP connection object
http% = getdata%

; connect to the given site
NS_HTTP_CONNECT http%, URL
NS_HTTP_ADDREQUESTPROPERTY http%,"Accept-Language","fr"
NS_HTTP_SET_FOLLOWREDIRECT http%, FALSE%
buf$ = "NOM="&NOM&"&B1=Envoyer"
NS_HTTP_POSTEX http%, buf$, "application/x-www-form-urlencoded"
IF (NS_HTTP_ERROR <> 0)
   ERROR = NS_HTTP_GET_ERRORMSG
ENDIF
INSERT AT END "Statut =
"&&NS_HTTP_GET_STATUSCODE(http%)&&NS_HTTP_GET_REASONPHRASE(http%) to MLE

; get all header fields and add them to the text area
S1.TEXT= NS_HTTP_GET_HEADEREX (http%)

; read the content of the http response and write it to the text area or to a
; file
    S2.text = NS_HTTP_READEX(http%)
IF (NS_HTTP_ERROR <> 0)
   ERROR.TEXT = NS_HTTP_GET_ERRORMSG
ENDIF

; free the http connection object later
```

See also NS_HTTP_GET_HEADER, NS_HTTP_GETLINEEX, NS_HTTP_POSTEX, NS_HTTP_READEX

NS_HTTP_GETLINEEX function

Return a line of the HTTP flow as DynStr.

Syntax **NS_HTTP_GETLINEEX** (*pConnection*)

Parameter *pConnection* POINTER I connection's handle

Return value DynStr

Example

```
LOCAL POINTER http%
LOCAL I%
LOCAL Dynstr line$

S1 = ""
S2 = ""

; get HTTP connection object
http% = getdata%

; connect to the given site
NS_HTTP_CONNECT http%, URL

NS_HTTP_GET http%

IF (NS_HTTP_ERROR <> 0)
   ERROR = NS_HTTP_GET_ERRORMSG
ENDIF
INSERT AT END "Statut =
"&&NS_HTTP_GET_STATUSCODE(http%)&&NS_HTTP_GET_REASONPHRASE(http%) to MLE
S1.TEXT= NS_HTTP_GET_HEADEREX (http%)
; read the content of the http response and write it to the text area or to a
; file
WHILE (NS_HTTP_EOF(http%) = 0)
    line$ = NS_HTTP_GETLINEEX(http%)
    INSERT AT END line$ TO S2
ENDWHILE

IF (NS_HTTP_ERROR <> 0)
   ERROR = NS_HTTP_GET_ERRORMSG
ENDIF
; free the http connection object later
```

See also NS_HTTP_GETLINE, NS_HTTP_GET_HEADEREX, NS_HTTP_POSTEX, NS_HTTP_READEX

NS_HTTP_POSTEX instruction

Send an HTTP query using the POST method. Unlike the GET method, data is sent in the body (after the CONTENT-LENGTH header) of the query and not with the URL.

Syntax **NS_HTTP_POSTEX** *pConnection, ds, contenttype$*

Parameters

pConnection	POINTER	I	connection's handle
ds	DYNSTR	I	contain the data to send
contenttype$	CSTRING	I	CONTENT-TYPE of data to send. For example "text/xml"

Notes

1. This instruction should be called after NS_HTTP_CONNECT.
2. For a POST query, the fields are transmitted behind the MIME CONTENT-LENGTH header, they are separated by the & character (not to be confused with the & of NCL which serves to concatenate strings), the spaces are replaced with +s, any accented characters and &, / ... are encoded and replaced by their ANSI code preceded by the % character (for example 'à' by %E0).
   On receiving the POST query, the target HTTP server will pass control to the procedure quoted as an argument by transmitting the form fields to it on its standard input. The process is unique to the POST method, in the case of the GET method, the fields are retrieved in an environment variable.

Example

```
Local dynstr URL$
local dynstr ds
Local dynstr dsRet
LOCAL pointer http%

URL$ = "http://nstest.natsys.fr/infos.php"
ds =
"Nom=NAT%20SYSTEM&Ville=LA%20ROCHELLE&Departement=17&Produit=NSDK&Version=V5"

; create HTTP connection object
http% = NS_HTTP_NEW

; connect to the given site
NS_HTTP_CONNECT http%, URL$

; Send data with POST Method
NS_HTTP_POSTEX http%, ds, "application/x-www-form-urlencoded"

IF (NS_HTTP_ERROR <> 0)
      INSERT AT END NS_HTTP_GET_ERRORMSG TO MLELOG
else
```

```
      insert at end "data succesfully send" to MLELOG
ENDIF

; read the content of the http response and write it to the text area
dsRet = NS_HTTP_READEX(http%)

IF (NS_HTTP_ERROR <> 0)
      INSERT AT END NS_HTTP_GET_ERRORMSG TO MLELOG
else
      MLELOG.TEXT = dsRet
ENDIF

; free the http connection object
NS_HTTP_DISPOSE(http%)
```

See also NS_HTTP_CONNECT, NS_HTTP_POST

NS_HTTP_READEX function

Returns the number of bytes read.

Syntax **NS_HTTP_READEX** (*pConnection*)

Parameters *pConnection* POINTER I connection's handle

Return value DynStr

Notes

1. This function allows the http flow to be read in a single pass. Iterations are no longer needed.
2. In order to be able to display a character string longer than 255 characters in a control use the Text dynamic attribute. See the example.

Example

```
LOCAL POINTER http%
LOCAL DynStr ds

; S1 and S2 are two MLEs
S1 = ""
S2 = ""

; create HTTP connection object
http% = NS_HTTP_NEW
; connect to the given site
NS_HTTP_CONNECT http%, URL
NS_HTTP_GET http%

IF (NS_HTTP_ERROR <> 0)
   ERROR = NS_HTTP_GET_ERRORMSG
ENDIF

; get all header fields and add them to the text area
S1.TEXT = NS_HTTP_GET_HEADEREX(http%)

; read the content of the http response and write it to the text area
ds = NS_HTTP_READEX(http%)
; by using the .text attribute we could transfer a string greater than 255
; characters to an MLE
S2.text = ds

IF (NS_HTTP_ERROR <> 0)
   ERROR = NS_HTTP_GET_ERRORMSG
ENDIF
; free the http connection object
NS_HTTP_DISPOSE(http%)
```

See also NS_HTTP_READ

NWXML Library

This library allows you to handle the XML files.

NWX_NODE_OUT_DYNSTR Function

Exports the content of a node to a dynamic string.

Syntax **NWX_NODE_OUT_DYNSTR** (*pNode)*

Parameter *pNode* POINTER I/O node pointer

Return value DYNSTR string corresponding to the document

Example

```
local pointer pDoc
local pointer pParser, pointer pRootNode
local buffer$
local DynStr dsNode
local ret%
; Initialize NWXML
ret% = NWX_INITIALIZE%
;New parser
pParser = NWX_PARSER_NEW
; load the buffer in pDoc and pParser
buffer$ = '<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes"?><AUTO><MODELE>Toyota</MODELE><ANNEE>1953</ANNE
E></AUTO>'
pDoc = NWX_DOCUMENT_GETFROMBUFFER(buffer$, pParser)
; Search the root node of the document
pRootNode = NWX_DOCUMENT_GETROOT (pDoc)
; display the node
dsNode = NWX_NODE_OUT_DYNSTR(pRootNode)
MLE1.TEXT = dsNode
; free the document
NWX_DOCUMENT_DISPOSE pDoc
; free the parser
NWX_PARSER_DISPOSE pParser
; free NWXML
NWX_TERMINATE
```

Résultat :

```
<AUTO><MODELE>Toyota</MODELE><ANNEE>1953</ANNEE></AUTO>
```

See also NWX_DOCUMENT_OUT_DYNSTR

NWX_DOCUMENT_OUT_DYNSTR Function

Exports the content of a XML document to a dynamic string.

Syntax	**NWX_DOCUMENT_OUT_DYNSTR** (*pDoc)*			
Parameter	*pDoc*	POINTER	I/O	document pointer
Return value	DYNSTR string corresponding to the document			
See also	NWX_NODE_OUT_DYNSTR			