NS-DK

Version 6.00 Edition 1

Services Libraries Reference

Volume 4

NS-DK APIs

Ref. n° NSR600SLR04001

Contents

About this manual

This is Volume 4 of the NS-DK "Service Libraries Reference". The purpose of the Service Libraries Reference is to describe in detail the contents of the standard add-on libraries for the NCL language and explain how to use them.

Organization of the manual

This manual contains three chapters.

Chapter 1	**NSXMLPRJ library** This chapter describes how to manipulate the NS-DK projects.
Chapter 2	**NSSEQ sequences library** This chapter describes the programming interface of the NSSEQ sequences library delivered with NS-DK.
Chapter 3	**User Functions Library NSFCTUTIL** This chapter describes the library of NSFCTUTIL user functions that are supplied with the NS-DK tool.

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

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

Operating conventions

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

Icon codes

☞	**Comment,** note, etc.
👓	**Reference** to another part of the documentation
⚠	**Danger**: precaution to be taken, irreversible action, etc.

💡	**Suggestion**: helpful hints, etc.
⇗	**To go a step further**: level of detail or expertise greater than the average level of the document
⊞	Indicates specific information on using the software under DOS-Windows (all versions)
⊞ 32	Indicates specific information on using the software under DOS-Windows 32 bits
32 bits	Indicates specific information on using the software under DOS-Windows 32 bits
UNIX	Indicates specific information on using the software under Unix systems

Chapter 1 NSXMLPRJ library

The NSXMLPRJ library is used by NS-DK in order to handle NS-DK projects, whatever their format (binary or XML).

You will find in this chapter

- The prerequisites for using the NSXMLPRJ library
- How NSXMLPRJ operates and how to use it

Table of contents

Prerequisites

1. In version 5, the NS-DK product includes as standard the three files required for developing applications handling NS-DK project description files.

   <NSDK>\LIB\NSW2PRJ.LIB
   <NSDK>\DLL\NSW2PRJ.DLL
   <NSDK>\NCL\NSXMLPRJ.NCL

   NB: <NSDK> represents the directory you chose when you installed NS-DK.

   The API's operate on both the binary type configuration files proprietary to Nat System (xxx.prj), NS-DK version 3 format and earlier, and on the XML type configuration files (xxx.xnp), the new format used in version 5 of NS-DK. Old files (.prj) are only accessible in read-only mode.

   The NSXMLPRJ.NCL APIs replace those provided by V150.NCL located in the **contrib** directory of versions 3 and earlier of NS-DK.

   No specific installation or configuration needs to be carried out therefore in order to access the functionality of the API's.

2. Don't forget to deliver the NSW2XPRJ.DLL library with the application produced if you are building deliverables using these functionalities.

The user could display the NSXMLPRJ.NCL file in order to check the list of constants used by the API'S (130 to date).

Introduction

The NSXMLPRJ library lets you handle NS-DK projects, whatever their format (binary or XML).

The API uses a project handle (pointer) (the *pPrj* parameter).

Each project attribute has a unique number. Some attributes are indexed, either in a fixed size table, or in a table the size of which is another attribute of the integer type.

The same table may be accessed by several types of attribute. (Their number depends on the same integer attribute). There are four types of attribute:

- Boolean (the integer 0 or the strings “no” or “false” are equivalent, the case is ignored, as is any non-null integer as well as the strings “yes” or “true”).
- Integer, which can be broken down into two sub-types, integers with maximum/minimum limits, and an enumeration index.
- String.
- Sub-project (currently, a single attribute, xpaGenConfig%, which is indexed and lets you access build configurations). This mode of operation lets you use more than one index level.

When an attribute is being written, the new value is validated (and possibly rejected).

Writing to an indexed attribute causes the creation of elements (values) up to the value of this index, if they do not already exist, as well as values (default) for dependent attributes (linked, they use the same integer attribute for the size of their tables). If an integer attribute gives the size of the table, it is updated if necessary.

When elements are added to a table of the enumeration type without being directly changed, these elements automatically take the default value of this enumeration.

In order to add elements to an indexed attribute (variable size table), we strongly recommend either starting with the largest index, or by first setting the attribute determining the size of the table, in order to avoid pointless reallocation.

If this introduction seems a bit arid due to its technical nature, please do not worry, an example supplied at the end of the documentation will help you understand APIs used in handling project configuration better.

This example should enable you to prepare programs that allow you to read or change the attributes and resources of your NS-DK projects as needed.

Don't forget that you can look up the xxx.xnp project configuration files which are in XML format using any text editor.

A learning phase will however be necessary in order to master the use of the APIs.

Indeed, there are only a few APIs, but they operate on more than a hundred variables, each representing an aspect of the configuration of a project.

Functional categories of the NSXMLPRJ library

There follows a list of the functionalities managed by NS-DK's NSXMLPRJ library.

Opening and saving pages

Managing project versions

Managing attributes

NSXMLPRJ Reference library

XPRJ_New Function

Creates a new empty NS-DK project.

Syntax	**XPRJ_New**
Return value	SEG_XPRJ@

Note

1. To free the project after its use with the XPRJ_Dispose instruction.

See also XPRJ_Dispose, SEG_XPRJ segment

XPRJ_Load Function

Loads a project and returns a pointer to it.

Syntax **XPRJ_Load** (*file$*, *err*)

Parameters	*file$*	CSTRING	I	name of the project
	err	SEGMENT	O	SEG_XPrjErr segment

Return value SEG_XPRJ@

Notes

1. To free the project after its use with the XPRJ_Dispose instruction.
2. If the reference returned is null (loading error), *err* contains information on the nature of the error. For the moment, only the first three fields of this variable are filled in.
3. If the project is in PRJ format, the associated N_Cs are not loaded before there is an initial explicit request for access (XPRJ_LoadConfig or XPRJ_GetPtrAttrIdx%(pPrj, xpaGenConfig%, Index%, rpPrjCfg)).

Example

```
Local SEG_XPRJ@ pPrj, SEG_XPrjErr err

; Loads the project into memory from a .PRJ file, the .N_Cs (+ .CFG ?)
; assigns the reference returned to the variable pPrj
@pPrj = XPRJ_Load("ADE_DSGN.PRJ", err)
If err.error = 0 ; loading successful?
      ; Changes the Help Files attribute of the project in "NSDESIGN"
      If XPRJ_SetTextAttr%(pPrj, xpaOptHelpfiles%, "NSDESIGN")
             XPRJ_Save pPrj, "ade_dsgn.xnp", 1 ; Saves in XML format
      EndIf
      XPRJ_Dispose pPrj ; restores the memory used
EndIf
```

See also SEG_XPrjErr segment, XPRJ_LoadConfig, XPRJ_GetPtrAttrIdx%, XPRJ_Dispose

XPRJ_LoadConfig Function

Returns the sub-project corresponding to the generation configuration with the name *name$*.

Syntax	**XPRJ_LoadConfig** (*pPrj*, *name$*, *err*)			
Parameters	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
	name$	CSTRING	I	name of the generation configuration
	err	SEGMENT	O	SEG_XPrjErr segment
Return value	SEG_XPRJ@			

Notes

1. To free the sub-project after its use with the XPRJ_Dispose instruction. However, its data (even changed) remains in memory as long as the project to which the configuration belongs is itself not released.

2. If the reference returned is null (loading error, only if the configuration is in an N_C and this is the first access since the project was loaded), *err* contains information on the nature of the error. For the moment, only the first three fields of this variable are filled in.

3. If the project is in binary format (.PRJ) and the name of the configuration is not part of the project, but an N_C file with the requested name is found, this configuration is automatically added to the project. This is not the case for .XNP projects since the configurations are in the project file.

4. If an '*' is added before the configuration name (character prohibited in configuration names which must be valid filenames), this indicates that the configuration must be created if it does not yet exist (with default values). Failing this, an error is returned.

See also XPRJ_Dispose, SEG_XPRJ and SEG_XPrjErr segments, XPRJ_Load

XPRJ_Dispose Instruction

Frees a project obtained using the following APIs: XPRJ_New, XPRJ_Load, XPRJ_LoadConfig, XPRJ_GetPtrAttr% or XPRJ_GetPtrAttrIdx% if the attribute obtained is a sub-project (currently, only xpaGenConfig% with XPRJ_GetPtrAttrIdx%), in this case or in that of XPRJ_LoadConfig, the values associated with the sub-project are not lost as long as the parent project is retained, it's only the access handle to the sub-project that is released (but the call is compulsory in order to release the memory allocated by XPRJ_GetPtrAttrIdx%(xpaGenConfig%, …) or by XPRJ_LoadConfig).

Syntax	**XPRJ_Dispose** *pXPrj*			
Parameter	*pXPrj*	SEGMENT	I	SEG_XPRJ@ segment

Example

```
Local SEG_XPRJ@ pPrj, SEG_XPrjErr err

; Loads the project into memory from a .PRJ file, the .N_Cs (+ .CFG ?)
; assigns the reference returned to the variable pPrj
@pPrj = XPRJ_Load("ADE_DSGN.PRJ", err)
If @pPrj ; loading successful?
      ; Changes the Help Files attribute of the project in "NSDESIGN"
      If XPRJ_SetTextAttr%(pPrj, xpaOptHelpfiles%, "NSDESIGN")
             XPRJ_Save pPrj, "ade_dsgn.xnp", 1 ; Saves in XML format
      EndIf
      XPRJ_Dispose pPrj ; restores the memory used
EndIf
```

See also SEG_XPRJ and SEG_XPrjErr segments, XPRJ_New, XPRJ_Load, XPRJ_LoadConfig, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%, XPRJ_Dispose, XPRJ_Load

XPRJ_IsChanged% Function

Indicates whether a project has been changed since its creation, its loading or the last successful XPRJ_Save%.

Syntax	**XPRJ_IsChanged%** (*pXPrj*)			
Parameter	*pXPrj*	SEGMENT	O	SEG_XPRJ@ segment
Return value	INT(1)			
See also	XPRJ_Save%, SEG_XPRJ segment			

XPRJ_Save% Function

Saves a project.

Syntax	**XPRJ_Save%** (*pPrj*, *file$*, *version%*)		
Parameters	*pPrj*	SEGMENT I	SEG_XPRJ@ segment
	file$	CSTRING I	name of the project
	version$	INTEGER I	format of the project
Return value	INT(1)		

Notes

1. Do not use this function with a sub-project (generation configuration).
2. The xpsPrjVer% attribute retains the format of a loaded project, 0 : binary .PRJ and associated files, 1 or more: XML, upgradeable. The xpsLatestVer% attribute keeps the very latest version.
3. If the project was loaded using the XPRJ_Load function or already saved with a name by another call to XPRJ_Save%.
4. The *file$* parameter may be empty, in this case the former name is retained and the file(s) is/are are only written if at least one element of data has changed (by calling an XPRJ_SetXxx **with a different value from the original value** in an attribute, even if subsequently the original value is restored).
5. In the case of binary format, the names of the .N_Cs depend on the names of the corresponding generation configurations, they must therefore be valid file names.
6. Currently, you cannot use a version number lower than that of the current project.

Example

```
Local SEG_XPRJ@ pPrj, SEG_XPrjErr err

; Loads the project into memory from a .PRJ file, the .N_Cs (+ .CFG ?)
; assigns the reference returned to the variable pPrj
@pPrj = XPRJ_Load("ADE_DSGN.PRJ", err)
If @pPrj ; loading successful?
      ; Changes the Help Files attribute of the project in "NSDESIGN"
      If XPRJ_SetTextAttr%(pPrj, xpaOptHelpfiles%, "NSDESIGN")
             XPRJ_Save pPrj, "ade_dsgn.xnp", 1 ; Saves in XML format
      EndIf
      XPRJ_Dispose pPrj ; restores the memory used
EndIf
```

See also XPRJ_Load, SEG_XPRJ segment

XPRJ_OrgVersion% Function

Lets you set the project version.

Syntax	**XPRJ_OrgVersion%** *(pPrj)*			
Parameter	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
See also	XPRJ_Save%, XPRj_MinVersion%, SEG_XPRJ segment			

XPRJ_MinVersion% Function

Sets the minimum version of the format in which the project can be saved (if extensions were used as resource names with more than eight characters, new types of resources). **Not currently implemented.**

Syntax	**XPRj_MinVersion%** *(pPrj)*			
Parameter	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
Return value	INTEGER			
See also	XPRJ_OrgVersion%, SEG_XPRJ segment			

XPRJ_GetBoolAttr% Function

Retrieves the value of a Boolean attribute

Syntax **XPRJ_GetBoolAttr%** *(pPrj, Attrib%, Val%)*

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Val%	INT(1)	O	value of a boolean attribute

Return value INT(1)

value	**meaning**
false%	failure to retrieve the value.
non false%	success in retrieving the value.

See also xpa*% constants, SEG_XPRJ segment, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_GetBoolAttrIdx% Function

Retrieves the value of an indexed attribute.

Syntax **XPRJ_GetBoolAttrIdx%** *(pPrj, Attrib%, Index%, Val%)*

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Index%	INTEGER	I	index
Val%	INT(4)	O	value of an indexed attribute

Return value INT(1)

value	**meaning**
false%	failure to retrieve the value.
non false%	success in retrieving the value.

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_SetBoolAttr% Function

Changes the value of a Boolean attribute.

Syntax **XPRJ_SetBoolAttr%** *(pPrj, Attrib%, Val%)*

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Val%	INT(4)	I	value of a boolean attribute

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_SetBoolAttrIdx% Function

Changes the value of an indexed Boolean attribute.

Syntax **XPRJ_SetBoolAttrIdx%** *(pPrj, Attrib%, Index%, Val%)*

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Index%	INTEGER	I	index
Val%	INT(4)	I	value of a boolean attribute

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_GetIntAttr% Function

Retrieves the value of an integer or enumerated attribute.

Syntax **XPRJ_GetIntAttr%** (*pPrj*, *Attrib%*, *Val%*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Val%	INT(4)	O	attribute value

Return value INT(1)

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_GetIntAttrIdx% Function

Retrieves the value of an indexed integer or enumerated attribute.

Syntax **XPRJ_GetIntAttrIdx%** (*pPrj*, *Attrib%*, *Index%*, *Val%*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Index%	INTEGER	I	index
Val%	INT(4)	O	attribute value

Return value INT(1)

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_SetIntAttr% Function

Changes the value of an integer or enumerated attribute.

Syntax **XPRJ_SetIntAttr%** (*pPrj*, *Attrib%*, *Val%*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Val%	INT(4)	O	attribute value

Return value INT(1)

Note

1. Limits are imposed by the attribute. If the attribute (of necessity an integer in this case) determines the size of one or more indexed attribute tables, this is redimensioned in consequence (it is the end of the table(s) that changes).

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_SetIntAttrIdx% Function

Changes the value of an indexed integer or enumerated attribute.

Syntax **XPRJ_SetIntAttrIdx%** (*pPrj*, *Attrib%*, *Index%*, *Val%*)

Parameters	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
	Attrib%	INTEGER	I	xpa*% constants
	Index%	INTEGER	I	index
	Val%	INT(4)	I	attribute value

Return value INT(1)

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_GetTextAttr% Function

Returns the value of a string type attribute.

Syntax **XPRJ_GetTextAttr%** (*pPrj*, *Attrib%*, *Val$*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Val$	CSTRING	O	attribute value

Return value INT(1)

Note

1. This function returns the value of a string type attribute with a maximum of 255 characters. Use XPRJ_GetPtrAttr% for longer strings.

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_GetTextAttrIdx% Function

Returns the value of an indexed attribute.

Syntax **XPRJ_GetTextAttrIdx%** (*pPrj*, *Attrib%*, *Index%*, *Val$*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Index%	INTEGER	I	index
Val$	CSTRING	O	attribute value

Return value INT(1)

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_SetTextAttr% Function

Changes the value of **any type of attribute**. If the type is not a string, the value is SKIPPED.

Syntax **XPRJ_SetTextAttr%** (*pPrj*, *Attrib%*, *Val$*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Val$	CSTRING	O	attribute value

Return value INT(1)

Note

1. The table below presents behavior according to attribute type:

Type	Validation
Boolean	The string must either be any valid integer (FALSE% for 0, TRUE% for any other value) or one of the following strings (without taking account of the case and without the quotation marks): "false", "no", "true" or "yes".
Enumeration	The string must either be a valid integer between 0 and the number of elements in the enumeration - 1, or one of the enumeration values in letters and ignoring the case.
Integer	The string must be a valid integer between the limits permitted by the attribute.

Integers are in NCL format ('-'?([0-9]+|'$'[0-9A-Fa-f]+)).

Example

```
Local SEG_XPRJ@ pPrj, SEG_XPrjErr err

; Loads the project into memory from a .PRJ file, the .N_Cs (+ .CFG ?)
; assigns the reference returned to the variable pPrj
@pPrj = XPRJ_Load("ADE_DSGN.PRJ", err)
If @pPrj ; loading successful?
      ; Changes the Help Files attribute of the project in "NSDESIGN"
      If XPRJ_SetTextAttr%(pPrj, xpaOptHelpfiles%, "NSDESIGN")
            XPRJ_Save pPrj, "ade_dsgn.xnp", 1 ; Saves in XML format
      EndIf
      XPRJ_Dispose pPrj ; restores the memory used
EndIf
```

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_SetTextAttrIdx% Function

Changes the value of **any type of indexed attribute**. If the type is not a string, the value is SKIPPED.

Syntax **XPRJ_SetTextAttrIdx%** (*pPrj*, *Attrib%*, *Index%*, *Val$*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Index%	INTEGER	I	index
Val$	CSTRING	I	attribute value

Return value INT(1)

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_GetPtrAttr% Function

Returns a pointer to a sub-project (can be used as the first parameter *pPrj* with the other API functions) for a complex attribute.

Syntax **XPRJ_GetPtrAttr%** (*pPrj*, *Attrib%*, *pVal*)

Parameters

pPrj	SEGMENT	I	segment SEG_XPRJ@
Attrib%	INTEGER	I	constants xpa*%
pVal	SEGMENT	O	segment SEG_PXPRJ

Return value INT(1)

Notes

1. To free the project after its use with the XPRJ_Dispose instruction.
2. This function can also retrieve a pointer to CSTRING for string type attributes (without a length limit), the text to which the pointer refers must be used immediately before any action to change the project occurs and the pointer must not be retained (and no XPRJ_Dispose in this case). You should use a special segment containing a reference to a CString, a cast is required:

```
SEGMENT SEG_CStrRef
      CString@ S($FF00)
ENDSEGMENT ; SEG_CStrRef
…
Local SEG_CStrRef rs
…
If XPRJ_GetPtrAttr%(pPrj, Attrib%, SEG_PXPRJ(@rs))
      ; use of rs.S
EndIf
```

See also xpa*% constants, SEG_XPRJ segment, SEG_PXPRJ, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttrIdx%

XPRJ_GetPtrAttrIdx% Function

Returns a pointer to the attributes of a sub-project and indexed string (can be used as the first parameter *pPrj* with the other API functions) for a complex attribute.

Syntax **XPRJ_GetPtrAttrIdx%** (*pPrj*, *Attrib%*, *Index%*, *pVal*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Index%	INTEGER	I	index
pVal	SEGMENT	O	SEG_PXPRJ segment

Return value INT(1)

Example

```
Local SEG_PXPRJ rpPrj, Integer Index%, Integer Cnt%, Name$
…
; as XPRJ_GetIntAttr%(pPrj, xpaGenCount%, Cnt%)
Cnt% = XPRJ_Count%(pPrj, xpaGenConfig%) ; mais plus lisible !
For Index% = 0 To Cnt% - 1
      If XPRJ_GetPtrAttrIdx%(pPrj, xpaGenConfig%, Index%, rpPrj) \
      And @rpPrj.pPrj ; pointer paranoiac test!
             ; Using rpPrj.pPrj instead of  pPrj
             ; with xpaCfgName%...xpaCfgResSrcIndex% attributes
             XPRJ_SetBoolAttr% rpPrj.pPrj, xpaCfgHelpInfo%, TRUE%
             XPRJ_Dispose rpPrj.pPrj
      EndIf
EndFor
XPRJ_Save% pPrj, "", xpsPrjVer%
```

See also xpa*% constants, SEG_XPRJ and SEG_PXPRJ segments**,** XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%

Def* Functions

Def* functions are used in the same way as the others, they allow the same attributes to be read.

The differences are:

- There is no return code sent, the value read is returned by the function.
- The default value must be passed as the last argument of the function. If the call to the function fails, it's this last parameter that will be returned.

These functions can be used in place of the others at the developer's convenience.

XPRJ_DefGetBoolAttr% Function

Retrieves the value of a Boolean attribute

Syntax	**XPRJ_DefGetBoolAttr%** *(pPrj, Attrib%, Def%)*			
Parameters	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
	Attrib%	INTEGER	I	xpa*% constants
	Def%	INT(4)	I	default value
Return value	INT(1)			
	Value of the attribute considered in case of success, otherwise the default value indicated in the parameter *Def%*.			
See also	xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%			

XPRJ_DefGetBoolAttrIdx% Function

Retrieves the value of a Boolean attribute

Syntax	**XPRJ_DefGetBoolAttrIdx%** *(pPrj, Attrib%, Index%, Def%)*			
Parameters	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
	Attrib%	INTEGER	I	xpa*% constants
	Index%	INTEGER	I	index
	Def%	INT(4)	I	default value
Return value	INT(1)			
	Value of the attribute considered in case of success, otherwise the default value indicated in the parameter *Def%*.			

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_DefGetIntAttr% Function

Retrieves the value of an integer or enumerated attribute.

Syntax	**XPRJ_DefGetIntAttr%** (*pPrj*, *Attrib%*, *Def%*)			
Parameters	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
	Attrib%	INTEGER	I	xpa*% constants
	Def%	INT(4)	I	default value
Return value	INT(4)			
	Value of the attribute considered in case of success, otherwise the default value indicated in the parameter Def%.			
See also	xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%			

XPRJ_DefGetIntAttrIdx% Function

Retrieves the value of an indexed integer or enumerated attribute.

Syntax **XPRJ_DefGetIntAttrIdx%** (*pPrj*, *Attrib%*, *Index%*, *Def%*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Index%	INTEGER	I	index
Def%	INT(4)	I	default value

Return value INT(4)

Value of the attribute considered in case of success, otherwise the default value indicated in the parameter Def%.

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_DefGetTextAttr$ Function

Returns the value of a string type attribute.

Syntax **XPRJ_DefGetTextAttr$** (*pPrj*, *Attrib%*, *Def$*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Def$	CSTRING	I	default value

Return value CSTRING@

Value of the attribute considered in case of success, otherwise the default value indicated in the parameter Def%.

Note

1. This function returns the value of a string type attribute with a maximum of 255 characters. Use XPRJ_GetPtrAttr% for longer strings.

See also xpa*% constants, SEG_XPRJ segment, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_GetTextAttrIdx%, XPRJ_DefGetTextAttrIdx$, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_DefGetTextAttrIdx$ Function

Returns the value of an indexed attribute.

Syntax **XPRJ_DefGetTextAttrIdx$** (*pPrj*, *Attrib%*, *Index%*, *Def$*)

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
Index%	INTEGER	I	index
Def$	CSTRING	I	default value

Return value CSTRING@

Value of the attribute considered in case of success, otherwise the default value indicated in the parameter Def%.

See also xpa*% constants, XPRJ_GetBoolAttr%, XPRJ_DefGetBoolAttr%, XPRJ_GetBoolAttrIdx%, XPRJ_DefGetBoolAttrIdx%, XPRJ_SetBoolAttr%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_SetBoolAttrIdx%, XPRJ_GetIntAttr%, XPRJ_DefGetIntAttr%, XPRJ_GetIntAttrIdx%, XPRJ_DefGetIntAttrIdx%, XPRJ_SetIntAttr%, XPRJ_SetIntAttrIdx%, XPRJ_GetTextAttr%, XPRJ_DefGetTextAttr$, XPRJ_GetTextAttrIdx%, XPRJ_SetTextAttr%, XPRJ_SetTextAttrIdx%, XPRJ_GetPtrAttr%, XPRJ_GetPtrAttrIdx%

XPRJ_NewIdx% Function

Lets you add elements (number specified in the parameter *Count%*) from an index (parameter *At%*) in the table of an indexed attribute.

Syntax **XPRJ_NewIdx%** *(pPrj, Attrib%, At%, Count%)*

Parameters

pPrj	SEGMENT	I	SEG_XPRJ@ segment
Attrib%	INTEGER	I	xpa*% constants
At%	INTEGER	I	index
Count%	INTEGER	I	number of elements to add from the *At%* index

Return value INTEGER

Note

1. If the index *At%* is negative, the insertion is made at the end of the table (the exact value of *At%* is then ignored). If several attributes share the same index, they are all changed in the same way. The corresponding integer attribute containing the number of elements is updated.

See also XPRJ_DelIdx%

XPRJ_DelIdx% Function

Lets you delete elements (number specified in the parameter *Count%*) from an index (parameter *At%*) in the table of an indexed attribute.

Syntax	**XPRJ_DelIdx%** *(pPrj, Attrib%, At%, Count%)*			
Parameters	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
	Attrib%	INTEGER	I	xpa*% constants
	At%	INTEGER	I	index
	Count%	INTEGER	I	number of elements to delete from the *At%* index
Return value	INT(1)			

Note

1. If several attributes share the same index, the elements in the same indices are deleted in all linked tables. The corresponding integer attribute containing the number of elements is updated.

See also XPRJ_NewIdx%

XPRJ_Count% Function

Returns the number of elements of an indexed attribute (identical to retrieving the attribute containing the corresponding number of elements).

Syntax	**XPRJ_Count%** *(pPrj, Attrib%)*			
Parameters	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
	Attrib%	INTEGER	I	xpa*% constants
Return value	INTEGER			
See also	XPRJ_NewIdx%, XPRJ_DelIdx%			

XPRJ_Parent% Function

Returns the parent project of a sub-project (obtained via the functions XPRJ_GetPtrAttr% or XPRJ_GetPtrAttrIdx%). Returns 0 if the project is not a sub-project (but *pPrj* must be a pointer to a valid project).

Syntax	**XPRJ_Parent%** *(pPrj)*			
Parameter	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
Return value	POINTER			
See also	XPRJ_GetPtrAttrIdx%, XPRJ_GetPtrAttr%			

XPRJ_SortRes% Function

Sorts the project resources (by name, indexed attribute xpaResName%) and is able therefore to change the order of the indexed attributes xpaResKind%, xpaResName% and xpaResTrgSet% (but they remain grouped together). The project is however not marked as changed and is called automatically at the end of XPRJ_Load and at the beginning of XPRJ_Save%.

Syntax	**XPRJ_SortRes%** *(pPrj)*			
Parameter	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
Return value	INT(1)			
See also	xpa*% constants, XPRJ_Load, XPRJ_Save%			

XPRJ_GetKindExts$ Function

Returns a list of extensions supported for a given type of resource, possibly separated by semi-colons.

Syntax **XPRJ_GetKindExts$** *(pPrj, xpelk%, DefOnly%)*

Parameters	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
	xpelk%	INTEGER	I	one of the xpelk*% constants, except xpelkInvalid%
	DefOnly%	INTEGER	I	indicates that you only want to obtain the default extension (the one used if the name of the resource does not include an extension).

Return value CSTRING@

Notes

1. Extensions do not start with a full stop.
2. Currently, only resources of the xpelkBMP% type support more than one extension (for Bitmap, GIF and JPEG files).

See also xpelk*% constants, SEG_XPRJ

XPRJ_MaxTextAttrLen% Function

For an attribute of the text type (indexed or not), returns the maximum number of characters.

Syntax	**XPRJ_MaxTextAttrLen%** *(pPrj, Attrib%)*			
Parameters	*pPrj*	SEGMENT	I	SEG_XPRJ@ segment
	Attrib%	INTEGER	I	xpa*% constants
Return value	INTEGER			
Note				

1. Warning, the development environment may impose different limits (26 characters instead of 31 for a resource name for example.)

See also xpa*% constants, SEG_XPRJ

xpa*% Constants

Indicate the attribute indices for the parameters *Attrib%* of the APIs XPRJ_xxx.

Syntax	**xpaInvalid%**
	xpaCtxPrimaryWindow%
	xpaCtxCurInterface%
	xpaCtxStartExeName%
	xpaCtxStartExeParms%
	xpaCtxStartExeMode%
	xpaCtxStartExeMini%
	xpaCtxStartExeNoClose%
	xpaCtxStartExeCurDir%
	xpaOptAutoSave%
	xpaOptAutoSaveDuration%
	xpaOptSaveBeforeRun%
	xpaOptMinimizeOnRun%
	xpaOptCUAConformance%
	xpaOptDisplayGrid%
	xpaOptHorzGrid%
	xpaOptVertGrid%
	xpaOptDesignMode%
	xpaOptDevelopmentMode%
	xpaOptDisplayBackground%
	xpaOptIgnoreTab%
	xpaOptTabSize%
	xpaOptBkMode%
	xpaOptBkName%
	xpaOptBkColor%
	xpaOptDlgBoxTemplate%
	xpaOptWindowTemplate%
	xpaOptHelpfiles%
	xpaOptLanguage%

xpaOptUndeclaredID%

xpaOptPrnIsText%

xpaOptPrnLines%

xpaOptPrnColumns%

xpaOptPrnFont%

xpaOptPrnTopLines%

xpaOptPrnLeftColumns%

xpaOptPrnDevice%

xpaOptPrnFormFeed%

xpaOptPrnInit%

xpaOptPrnHeader%

xpaOptConfigIO%

xpaOptLocks%

xpaDirExe%

xpaDirTmp%

xpaDirBak%

xpaDirScr%

xpaDirsScr%

xpaDirTpl%

xpaDirsTpl%

xpaDirSeg%

xpaDirsSeg%

xpaDirBmp%

xpaDirsBmp%

xpaDirLst%

xpaDirsLst%

xpaDirInc%

xpaDirsInc%

xpaDirLib%

xpaDirsLib%

xpaDirVar%

xpaDirsVar%

xpaDirRoot%

xpaEvtState%

xpaEvtX%

xpaEvtY%

xpaEvtW%

xpaEvtH%

xpaEvtFontName%

xpaEvtFontSize%

xpaEvtFontSels%

xpaEvtSelection%

xpaEvtSearchPattern%

xpaEvtWholeWord%

xpaEvtMatchCase%

xpaEvtFgCol%

xpaEvtBkCol%

xpaTrgCount%

xpaTrgGeneric%

xpaTrgCurrent%

xpaTrgName%

xpaTrgMode%

xpaTrgChSet%

xpaTrgScrSz%

xpaGenCount%

xpaGenConfig%

xpaResCount%

xpaResKind%

xpaResName%

xpaResTrgSet%

xpaCfgName%

xpaCfgMainScreen%

xpaCfgFarData%

xpaCfgHelpInfo%

xpaCfgSymbolsInfo%

xpaCfgMaxLineSize%

xpaCfgMaxLinesPerFile%

xpaCfgMaxIdLength%

xpaCfgTabSize%

xpaCfgExeName%

xpaCfgCPath%

xpaCfgCOptions%

xpaCfgLPath%

xpaCfgLPathImpLib%

xpaCfgLOptBin%

xpaCfgLOptDll%

xpaCfgLLibBin%

xpaCfgLLibDll%

xpaCfgLStackSize%

xpaCfgLHeapSize%

xpaCfgSourcesPath%

xpaCfgIncludesPath%

xpaCfgObjectsPath%

xpaCfgBinariesPath%

xpaCfgDLLsPath%

xpaCfgLibsPath%

xpaCfgNbSources%

xpaCfgSources%

xpaCfgDefaults%

xpaCfgDefault%

xpaCfgTrgCnt%

xpaCfgTrgNames%

xpaCfgRCPath%

xpaCfgRCOptions%

xpaCfgToolSetName%

xpaCfgGenOptions%

xpaCfgImpLibOptions%

xpaCfgResCount%

xpaCfgResKind%

xpaCfgResName%

xpaCfgResSrcIndex%

xpaCfgStandaloneExeName%

Notes

1. These constants are used in the functions and instructions XPRJ_xxx. They have the following meaning :

   Attributes indexes for the Attrib% parameters of the XPRJ_xxx API

   xpa = Xml Project Attribute

   for Enumerations, valid string values are separated by / and correspond

   to 0...n-1 integer values. The default value is surronded by ><.

xpaInvalid%	doesn't exist, don't pass to XPRJ_*
NSDESIGN options	
xpaCtxPrimaryWindow%	Text: the MainWindow name
xpaCtxCurInterface%	Enum: >Graph</Char/DosForm/Unknown
Start exe options	
xpaCtxStartExeName%	Text: The name of the executable to start
xpaCtxStartExeParms%	Text: the executable command line params
xpaCtxStartExeMode%	Int
xpaCtxStartExeMini%	Bool: minimize NSDesign when running exe
xpaCtxStartExeNoClose%	Bool
xpaCtxStartExeCurDir%	Text: active directory when the exe starts
NSDESIGN options	
xpaOptAutoSave%	Bool: auto-backup activation
xpaOptAutoSaveDuration%	Int : auto-backup time interval (seconds)
xpaOptSaveBeforeRun%	Bool: auto-save before run/generation
xpaOptMinimizeOnRun%	Bool: minimize NSDesign on debug/test
xpaOptCUAConformance%	Bool: CUA conformance
xpaOptDisplayGrid%	Bool: activate grid in dialogs edition
xpaOptHorzGrid%	Int : grid horizontal interval (pixels)
xpaOptVertGrid%	Int : grid vertical interval (pixels)
xpaOptDesignMode%	Bool: allow GUI edition
xpaOptDevelopmentMode%	Bool: allow code edition
xpaOptDisplayBackground%	deprecated Bool: transparent (False%)/opaque bkgnd
xpaOptIgnoreTab%	deprecated Bool: tabs or spaces in editor
xpaOptTabSize%	deprecated Int : tab size

xpaOptBkMode%	deprecated Enum: >NoBackground</Image/TiledImage/ScaledImage
xpaOptBkName%	deprecated Text: background bitmap filename
xpaOptBkColor%	deprecated Int : background color
xpaOptDlgBoxTemplate%	deprecated Text: ?
xpaOptWindowTemplate%	deprecated Text: ?
xpaOptHelpfiles%	Text: help filenames ; separated
xpaOptLanguage%	deprecated Enum: >C</Cobol/Pascal
xpaOptUndeclaredID%	Enum: Global/Local/>Error<
xpaOptPrnIsText%	Bool: direct text printing (=>escape codes)
xpaOptPrnLines%	Int : lines per page
xpaOptPrnColumns%	Int : columns per line
xpaOptPrnFont%	Text: fontname
xpaOptPrnTopLines%	Int : top margin (lines)
xpaOptPrnLeftColumns%	Int : left margin (characters)
xpaOptPrnDevice%	Text: printer device name
xpaOptPrnFormFeed%	Text: form feed control codes
xpaOptPrnInit%	Text: printer job initialisation codes
xpaOptPrnHeader%	Text[4]: pages header lines
xpaOptConfigIO%	deprecated Bool: uses .CFG
xpaOptLocks%	Int

Project resource directories

xpaDirExe%	Text: executable tools (gen/test) directory
xpaDirTmp%	Text: temporary files directory
xpaDirBak%	Text: backup files directory
xpaDirScr%	Text: SCR local resources directory
xpaDirsScr%	Text: SCR shared res. ; sep. dir list
xpaDirTpl%	Text: TPL local resources directory
xpaDirsTpl%	Text: TPL shared res. ; sep. dir list
xpaDirSeg%	Text: SEG local resources directory
xpaDirsSeg%	Text: SEG shared res. ; sep. dir list
xpaDirBmp%	Text: pictures local resources directory
xpaDirsBmp%	Text: pictures shared res. ; sep. dir list
xpaDirLst%	Text: TXT local resources directory

xpaDirsLst%	Text: TXT shared res. ; sep. dir list
xpaDirInc%	Text: included local NCL directory
xpaDirsInc%	Text: included shared NCL ; sep. dir list
xpaDirLib%	Text: NCL local resources directory
xpaDirsLib%	Text: NCL shared res. ; sep. dir list
xpaDirVar%	Text: VAR local resources directory
xpaDirsVar%	Text: VAR shared res. ; sep. dir list
xpaDirRoot%	deprecated Text

old editor options

xpaEvtState%	deprecated : Enum: editor window state >Null</Minimized/Maximized
xpaEvtX%	deprecated : Int : editor window X origin
xpaEvtY%	deprecated : Int : editor window Y origin
xpaEvtW%	deprecated : Int : editor window width
xpaEvtH%	deprecated : Int : editor window height
xpaEvtFontName%	deprecated : Text: editor window font name
xpaEvtFontSize%	deprecated : Int : editor window font size
xpaEvtFontSels%	deprecated : Int : editor window font attributes
xpaEvtSelection%	deprecated : Enum: list displayed resource kind >SCR</TPL/NCL/SEG/VAR/RPT/PTR/ICO/BMP/CST/TXT/BKS/BKT/ANY
xpaEvtSearchPattern%	deprecated : Text: last searched text
xpaEvtWholeWord%	deprecated : Bool: "whole word only" option
xpaEvtMatchCase%	deprecated : Bool: "case sensitive" option
xpaEvtFgCol%	deprecated : Int : editor window text color
xpaEvtBkCol%	deprecated : Int : editor window background color

Project infos

Targets

xpaTrgCount%	Int : targets count
xpaTrgGeneric%	Int : 1 based index of the generic target in xpaTrgName% array

xpaTrgCurrent%	Int : 1 based index of the current target in xpaTrgName% array
xpaTrgName%	Text[xpaTrgCount%]: target name
xpaTrgMode%	Enum[xpaTrgCount%]: >Graph</Char/DosForm/Unknown
xpaTrgChSet%	Enum[xpaTrgCount%]: ASCII/>ANSI</ASCIIMAC/EBCDIC/Unknown
xpaTrgScrSz%	deprecated Enum[xpaTrgCount%]: >VGA</XGA/Unknown

Generation configurations

xpaGenCount%	generation configuration count
xpaGenConfig%	deprecated : Sub-project[xpaGenCount] => XPRJ_GetPtrAttrIdx%

Project resources

xpaResCount%	Int : project resources count
xpaResKind%	Enum[xpaResCount%]: resource kind >SCR</TPL/NCL/SEG/VAR/RPT/PTR/ICO/BMP/CST/TXT/BKS/BKT/ANY
xpaResName%	Text[xpaResCount%]: resource name
xpaResTrgSet%	Int [xpaResCount%]: resources targets set (sum of 2 to the power of the zero based target index)

All attributes below are accessed through a child project pointer (xpaGenConfig), this allow sub-indexing

xpaCfgName%	Text: generation configuration name
xpaCfgMainScreen%	Text: the MainWindow screen name
xpaCfgFarData%	Bool: FARDATA generator option
xpaCfgHelpInfo%	Bool: HELPINFO generator option
xpaCfgSymbolsInfo%	Bool: generate with control names
xpaCfgMaxLineSize%	Int : maximum characters per line of generated source files
xpaCfgMaxLinesPerFile%	Int : maximum lines per generated source files
xpaCfgMaxIdLength%	Int : maximum NCL identifier length
xpaCfgTabSize%	Int : generated source files identation width
xpaCfgExeName%	Text: caller EXE filename when generating DLLs

xpaCfgCPath%	Text: C compiler path and filename
xpaCfgCOptions%	Text: C compiler command line options
xpaCfgLPath%	Text: linker path and filename
xpaCfgLPathImpLib%	Text: implib path and filename
xpaCfgLOptBin%	Text: linker command line options (for EXE)
xpaCfgLOptDll%	Text: linker command line options (for DLL)
xpaCfgLLibBin%	Text: libraries filenames (for EXE)
xpaCfgLLibDll%	Text: libraries filenames (for DLL)
xpaCfgLStackSize%	Int : stack size
xpaCfgLHeapSize%	deprecated Int : heap size
xpaCfgSourcesPath%	Text: directory for generated .C files
xpaCfgIncludesPath%	Text: directory for generated .D/.H files
xpaCfgObjectsPath%	Text: directory for generated .OBJ files
xpaCfgBinariesPath%	Text: directory for generated .EXE files
xpaCfgDLLsPath%	Text: directory for generated .DLL files
xpaCfgLibsPath%	Text: directory for generated .LIB files
xpaCfgNbSources%	Int : DLLs count
xpaCfgSources%	Text[xpaCfgNbSources%]: DLLs filenames
xpaCfgDefaults%	Int [13]: indexed by the same enum as the one used in xpaResKind%, contain indexes to ???
xpaCfgDefault%	Int : 1 based default target index in xpaCfgNames% array
xpaCfgTrgCnt%	Int : targets count
xpaCfgTrgNames%	Text[xpaCfgTrgCnt%]: targets names
xpaCfgRCPath%	Text: resource compiler path and filename
xpaCfgRCOptions%	Text: resource compiler command line options
xpaCfgToolSetName%	Text: appended after /TOOLKIND: generator command line option
xpaCfgGenOptions%	Text: others generator command line option
xpaCfgImpLibOptions%	Text: implib command line options
xpaCfgResCount%	Int : resource to EXE/DLLs mapping count
xpaCfgResKind%	Enum[xpaCfgResCount%]: >SCR</TPL/NCL/SEG/VAR/RPT/PTR/ICO /BMP/CST/TXT/BKS/BKT/ANY
xpaCfgResName%	Text[xpaCfgResCount%]: resource filename
xpaCfgResSrcIndex%	Int [xpaCfgResCount%]: 1 based index in xpaCfgSources% array

xpaCfgStandaloneExeName%	Text: the EXE name when not generating DLLs

2. Their internal declaration is:

```
xpaInvalid%             -1
xpaCtxPrimaryWindow%     0
xpaCtxCurInterface%      1
xpaCtxStartExeName%      2
xpaCtxStartExeParms%     3
xpaCtxStartExeMode%      4
xpaCtxStartExeMini%      5
xpaCtxStartExeNoClose%   6
xpaCtxStartExeCurDir%    7
xpaOptAutoSave%          8
xpaOptAutoSaveDuration%  9
xpaOptSaveBeforeRun%    10
xpaOptMinimizeOnRun%    11
xpaOptCUAConformance%   12
xpaOptDisplayGrid%      13
xpaOptHorzGrid%         14
xpaOptVertGrid%         15
xpaOptDesignMode%       16
xpaOptDevelopmentMode%  17
xpaOptDisplayBackground% 18
xpaOptIgnoreTab%        19
xpaOptTabSize%          20
xpaOptBkMode%           21
xpaOptBkName%           22
xpaOptBkColor%          23
xpaOptDlgBoxTemplate%   24
xpaOptWindowTemplate%   25
xpaOptHelpfiles%        26
xpaOptLanguage%         27
xpaOptUndeclaredID%     28
xpaOptPrnIsText%        29
xpaOptPrnLines%         30
xpaOptPrnColumns%       31
xpaOptPrnFont%          32
xpaOptPrnTopLines%      33
xpaOptPrnLeftColumns%   34
xpaOptPrnDevice%        35
xpaOptPrnFormFeed%      36
xpaOptPrnInit%          37
xpaOptPrnHeader%        38
xpaOptConfigIO%         39
xpaOptLocks%            40
xpaDirExe%              41
xpaDirTmp%              42
xpaDirBak%              43
xpaDirScr%              44
xpaDirsScr%             45
xpaDirTpl%              46
xpaDirsTpl%             47
xpaDirSeg%              48
xpaDirsSeg%             49
xpaDirBmp%              50
xpaDirsBmp%             51
xpaDirLst%              52
xpaDirsLst%             53
xpaDirInc%              54
xpaDirsInc%             55
xpaDirLib%              56
xpaDirsLib%             57
xpaDirVar%              58
xpaDirsVar%             59
```

```
xpaDirRoot%                 60

xpaEvtState%                61
xpaEvtX%                    62
xpaEvtY%                    63
xpaEvtW%                    64
xpaEvtH%                    65
xpaEvtFontName%             66
xpaEvtFontSize%             67
xpaEvtFontSels%             68
xpaEvtSelection%            69

xpaEvtSearchPattern%        70
xpaEvtWholeWord%            71
xpaEvtMatchCase%            72
xpaEvtFgCol%                73
xpaEvtBkCol%                74

xpaTrgCount%                75
xpaTrgGeneric%              76
xpaTrgCurrent%              77
xpaTrgName%                 78
xpaTrgMode%                 79
xpaTrgChSet%                80
xpaTrgScrSz%                81

xpaGenCount%                82
xpaGenConfig%               83

xpaResCount%                84
xpaResKind%                 85
xpaResName%                 86
xpaResTrgSet%               87
xpaCfgName%                 88
xpaCfgMainScreen%           89
xpaCfgFarData%              90
xpaCfgHelpInfo%             91
xpaCfgSymbolsInfo%          92
xpaCfgMaxLineSize%          93
xpaCfgMaxLinesPerFile%      94
xpaCfgMaxIdLength%          95
xpaCfgTabSize%              96
xpaCfgExeName%              97
xpaCfgCPath%                98
xpaCfgCOptions%             99
xpaCfgLPath%               100
xpaCfgLPathImpLib%         101
xpaCfgLOptBin%             102
xpaCfgLOptDll%             103
xpaCfgLLibBin%             104
xpaCfgLLibDll%             105
xpaCfgLStackSize%          106
xpaCfgLHeapSize%           107
xpaCfgSourcesPath%         108
xpaCfgIncludesPath%        109
xpaCfgObjectsPath%         110
xpaCfgBinariesPath%        111
xpaCfgDLLsPath%            112
xpaCfgLibsPath%            113
xpaCfgNbSources%           114
xpaCfgSources%             115
xpaCfgDefaults%            116
xpaCfgDefault%             117
xpaCfgTrgCnt%              118
xpaCfgTrgNames%            119
```

```
xpaCfgRCPath%             120
xpaCfgRCOptions%          121
xpaCfgToolSetName%        122
xpaCfgGenOptions%         123
xpaCfgImpLibOptions%      124
xpaCfgResCount%           125
xpaCfgResKind%            126
xpaCfgResName%            127
xpaCfgResSrcIndex%        128
xpaCfgStandaloneExeName% 129
```

See also xpei*%, xpel*%, xpeui*%, xpewpa*%, xpelk*%, xpees*%, xpecs*%, xpess*%, xps*% constants

xpei*% Constants

Indicate the types of interface used with the attributes xpaCtxCurInterface% and xpaTrgMode%.

Syntax

xpeiDefault%

xpeiGraph%

xpeiChar%

xpeiDosForm%

xpeiUnknown%

Note

1. Their declaration is the following:

```
xpeiDefault% 0
xpeiGraph%   0
xpeiChar%    1
xpeiDosForm% 2
xpeiUnknown% 3
```

See also

xpa*%, xpel*%, xpeui*%, xpewpa*%, xpelk*%, xpees*%, xpecs*%, xpess*%, xps*% constants

xpel*% Constants

Indicate the types of language used with the attributes xpaOptLanguage%.

Syntax

xpelDefault%

xpelC%

xpelCobol%

xpelPascal%

Note

1. Their declaration is the following:

```
xpelDefault% 0
xpelC%       0
xpelCobol%   1
xpelPascal%  2
```

See also xpa*%, xpei*%, xpeui*%, xpewpa*%, xpelk*%, xpees*%, xpecs*%, xpess*%, xps*% constants

xpeui*% Constants

Indicate the undeclared identifiers used with the attribute xpaOptUndeclaredID%.

Syntax

xpeuiDefault%

xpeuiGlobal%

xpeuiLocal%

xpeuiError%

Note

1. Their declaration is the following:

```
xpeuiDefault% 2
xpeuiGlobal%  0
xpeuiLocal%   1
xpeuiError%   2
```

See also

xpa*%, xpei*%, xpel*%, xpewpa*%, xpelk*%, xpees*%, xpecs*%, xpess*%, xps*% constants

xpewpa*% Constants

Indicate the undeclared identifiers used with the attribute xpaOptUndeclaredID%.

Syntax

xpewpaDefault%

xpewpaNoBkgnd%

xpewpaImage%

xpewpaTiledImage%

xpewpaScaledImage%

Note

1. Their declaration is the following:

```
xpewpaDefault%     0
xpewpaNoBkgnd%     0
xpewpaImage%       1
xpewpaTiledImage%  2
xpewpaScaledImage% 3
```

See also xpa*%, xpei*%, xpel*%, xpeui*%, xpelk*%, xpees*%, xpecs*%, xpess*%, xps*% constants

xpelk*% Constants

Indicate the types of resources used with the attributes xpaEvtSelection%, xpaResKind% and xpaCfgResKind%.

Syntax

xpelkInvalid%
xpelkTPL%
xpelkNCL%
xpelkSEG%
xpelkVAR%
xpelkRPT%
xpelkPTR%
xpelkICO%
xpelkBMP%
xpelkCST%
xpelkTXT%
xpelkBKS%
xpelkBKT%
xpelkMNU%
xpelkANY%

Note

1. Their declaration is the following:

```
xpelkInvalid% -1
xpelkSCR%      0
xpelkTPL%      1
xpelkNCL%      2
xpelkSEG%      3
xpelkVAR%      4
xpelkRPT%      5
xpelkPTR%      6
xpelkICO%      7
xpelkBMP%      8
xpelkCST%      9
xpelkTXT%     10
xpelkBKS%     11
xpelkBKT%     12
xpelkMNU%     13
xpelkANY%     14
```

See also xpa*%, xpei*%, xpel*%, xpeui*%, xpewpa*%, xpees*%, xpecs*%, xpess*%, xps*% constants

xpees*% Constants

Indicate the types of editing window used with the attribute xpaEvtState%.

Syntax

xpeesDefault%

xpeesNull%

xpeesMinimized%

xpeesMaximized%

Note

1. Their declaration is the following:

```
xpeesDefault%   0
xpeesNull%      0
xpeesMinimized% 1
xpeesMaximized% 2
```

See also xpa*%, xpei*%, xpel*%, xpeui*%, xpewpa*%, xpelk*%, xpecs*%, xpess*%, xps*% constants

xpecs*% Constants

Indicate the character sets used with the attribute xpaTrgChSet%.

Syntax

xpecsDefault%

xpecsASCII%

xpecsANSI%

xpecsASCIIMAC%

xpecsEBCDIC%

xpecsUnknown%

xpecsUTF8%

Note

1. Their declaration is the following:

```
xpecsDefault%  1
xpecsASCII%    0
xpecsANSI%     1
xpecsASCIIMAC% 2
xpecsEBCDIC%   3
xpecsUnknown%  4
xpecsUTF8%     255
```

See also xpa*%, xpei*%, xpel*%, xpeui*%, xpewpa*%, xpelk*%, xpees*%, xpess*%, xps*% constants

xpess*% Constants

Indicate the types of screens used with the attribute xpaTrgScrSz%.

Syntax

xpessDefault%

xpessVGA%

xpessXGA%

xpessUnknown%

Note

1. Their declaration is the following:

```
xpessDefault% 0
xpessVGA%     0
xpessXGA%     1
xpessUnknown% 2
```

See also xpa*%, xpei*%, xpel*%, xpeui*%, xpewpa*%, xpelk*%, xpees*%, xpecs*%, xps*% constants

xps*% Constants

Indicate the types of method of saving the NS-DK project.

Syntax **xpsPrjVer%**

xpsLatestVer%

Notes

1. Their meaning is as follows:

xpsPrjVer%	retains the current project version.
xpsLatestVer%	uses the latest version available

2. Their declaration is the following:

```
xpsPrjVer%    -1
xpsLatestVer% -2
```

See also xpa*%, xpei*%, xpel*%, xpeui*%, xpewpa*%, xpelk*%, xpees*%, xpecs*%, xpess*% constants

SEG_XPRJ Segment

Indicates the project handle.

Syntax	**SEGMENT SEG_XPRJ**
	INT dummy(1)
	ENDSEGMENT
Parameter	**dummy INT(1)**
See also	SEG_PXPRJ, SEG_XPrjErr segments

Segment SEG_PXPRJ

Syntax	**SEGMENT SEG_PXPRJ**
	SEG_XPRJ@ pPrj
	ENDSEGMENT
Parameter	**pPrj** **SEG_XPRJ@**
See also	SEG_XPRJ, SEG_XPrjErr segments

SEG_XPrjErr Segment

Segment used with the XPRJ_Load function in order to retrieve any errors.

Syntax

```
SEGMENT  SEG_XPrjErr
   INTEGER Error
   INTEGER Attrib
   INTEGER Index
   INTEGER Line
   INTEGER Column
   CSTRING Msg
ENDSEGMENT
```

Parameters

Error	**INTEGER**	**error code**
Attrib	**INTEGER**	**attribut where the error appears (if necessary)**
Index	**INTEGER**	**attribute index where the error appears (if necessary)**
Line	**INTEGER**	**error line**
Column	**INTEGER**	**error column**
Msg	**CSTRING**	**error message**

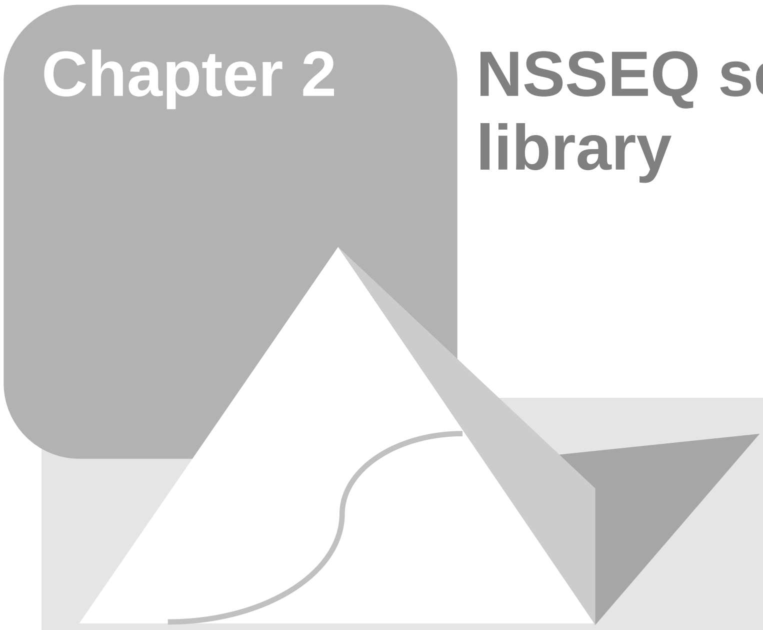
See also segments SEG_PXPRJ, SEG_XPRJ

Example of reading some project attributes

```
; ************************************************************
;  NSXMLPRJ.NCL use sample
; ************************************************************


local SEG_XPRJ@ pPrj, SEG_XPRJ@ pGenConfig
local SEG_XPrjErr err
local int ret% (1),u%,i%
local val$,val%,val2%
local nbres%, nbGen%
local configName$,ext$


; Trace function,
;Instruction TestTrace str$, control LISTBOX
;  insert at end str$ to LISTBOX
;EndInstruction


TestTrace "******** PROJECT READ BEGIN ********", LISTBOX

@pPrj=0
; ****************************
; Project loading
@pPrj = XPRJ_Load ("C:\projets\testv50.xnp", err)
if err.error <> 0
      message "Error","can't read project, error" && err.error
      exit
endif
; ****************************

TestTrace "",LISTBOX

TestTrace "------- RESOURCES LIST BEGIN--------", LISTBOX
ret% = XPRJ_getIntAttr% (pPrj, xpaResCount%, nbRes%)
TestTrace "Nb resources=" & nbRes%, LISTBOX

;;; les valeurs indexées commence à zéro
for i% = 0 to nbRes% - 1
  ret% = XPRJ_GetTextAttrIdx% (pPrj, xpaResName%,i%, val$)

  ret% = XPRJ_GetIntAttrIdx% (pPrj, xpaResKind%,i%, val%)
  ext$ = XPRJ_GetKindExts$ (pPrj,val%,false%)

  TestTrace "Resource (" && i% && ") =" & Val$ && ext$, LISTBOX
endFor
TestTrace "-------- RESOURCES LIST END --------", LISTBOX
TestTrace "",LISTBOX


TestTrace "-------- TARGET LIST BEGIN --------", LISTBOX
ret% = XPRJ_getIntAttr% (pPrj, xpaTrgCount%, nbRes%)
TestTrace "Nb Targets=" & nbRes%, LISTBOX

for i% = 0 to nbRes% - 1
  ret% = XPRJ_GetIntAttrIdx% (pPrj, xpaTrgChSet%,i%, val%)
  ret% = XPRJ_GetIntAttrIdx% (pPrj, xpaTrgMode%,i%, val2%)
  ret% = XPRJ_GetTextAttrIdx% (pPrj, xpaTrgName%,i%, val$)
  TestTrace "Targets (" && i% && ")= Name=" & Val$ && "Mode=" && Val% && "Code
Char=" && Val2%, LISTBOX
endFor
TestTrace "-------- TARGET LIST END --------", LISTBOX
TestTrace "",LISTBOX

TestTrace "-------- PROJECTS GENERATION BEGIN --------", LISTBOX
TestTrace "rem   :  a generation configuration is read as a subProject", LISTBOX

ret% = XPRJ_getIntAttr% (pPrj, xpaGenCount%, nbGen%)
TestTrace "Nb of generations=" && nbGen%, LISTBOX

for i% = 0 to nbGen% - 1
      ret% = XPRJ_GetTextAttrIdx% (pPrj, xpaCfgName%,i%, configName$)
      TestTrace "Build Configuration (" && i% && ")",LISTBOX
      TestTrace "   configName$=" & configName$,LISTBOX

  @pGenConfig = XPRJ_LoadConfig (pPrj, ConfigName$, err)

      ; generation options
      ret% = XPRJ_GetTextAttr% (pGenConfig, xpaCfgGenOptions%, val$)
      TestTrace "   Generation Options%=" & Val$, LISTBOX

      if pos%("/DLL",val$) = 0
             ; generated exe name
             ret% = XPRJ_GetTextAttr% (pGenConfig, xpaCfgStandaloneExeName%,
val$)
             TestTrace "   Exe name=" & Val$, LISTBOX
      else
             ; generated Dll names
             ret% = XPRJ_GetIntAttr% (pGenConfig, xpaCfgNbSources%, nbRes%)
             TestTrace "   nb DLL%=" & nbRes%, LISTBOX
             for u% = 0 to nbRes% -1
                    ret% = XPRJ_GetTextAttrIdx% (pGenConfig, xpaCfgSources%,
u%,val$)
                    TestTrace "      Dll ("& u% &") = " & Val$, LISTBOX
             EndFor
      endIf

      ret% = XPRJ_GetTextAttr% (pGenConfig, xpaCfgCPath%, val$)
      TestTrace "   C compiler path and filename=" & Val$, LISTBOX

      ret% = XPRJ_GetTextAttr% (pGenConfig, xpaCfgCOptions%, val$)
      TestTrace "   C compiler command line option=" & Val$, LISTBOX

      ret% = XPRJ_GetTextAttr% (pGenConfig, xpaCfgToolSetName%, val$)
      TestTrace "   TOOLKIND : " && "Val$=#" & Val$, LISTBOX

      ret% = XPRJ_GetTextAttr% (pGenConfig, xpaCfgSourcesPath%, val$)
      TestTrace "   Generated .C files directory=" & Val$, LISTBOX

      ret% = XPRJ_GetTextAttr% (pGenConfig, xpaCfgObjectsPath%, val$)
      TestTrace "   Generated .OBJ files directory =" & Val$, LISTBOX

endFor
TestTrace "-------- PROJECTS GENERATION END --------", LISTBOX



; save if modified
; ret% = XPRJ_SAVE% (pPrj,'d:\newprj.xml',1)
; TestTrace ("XPRJ_SAVE%" && ret%)


XPRJ_Dispose(pPrj)
```

```
TestTrace "******** Project READ END ********", LISTBOX
```

Chapter 2 NSSEQ sequences library



This chapter describes the programming interface of the NSSEQ sequences library delivered with NS-DK.

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

You will find in this chapter

- The description and syntax statement for each verb in the graphical builder libraries (NSSQEQ.NCL).
- The verbs used for client/server mode.

Table of contents

Installation and utilisation

After the installation of NS-DK, you must find NSSEQ.NCL in the <NSDK>\NCL directory, <NSDK> representing the directory specified during the installation.

Verify that NSSEQ.NCL is in the directory corresponding to \SERVICES\NCL.

During the generation as a DLL of your NS-DK components (libraries and classes), verify that NSSEQ.LIB is correctly specified in the *.DLL Libraries* and *.EXE Libraries* fields of the *Configurations* NS-DK dialog box in the *Compiler/Linker* tab.

To read the NSSEQ.NCL file

1. Go to the <NS-DK>\NCL directory.

   NB : <NS-DK> represents the directory you have chosen during the NS-DK installation.
2. Edit the NSSEQ.NCL file with any text editor.

Reference of the NSSEQ library

Here is the list of the functionalities allowing you to manage the client-server.

SEQ_ALLOC instruction

Allocates memory for a sequence.

Every allocated sequence must later be freed by means of a SEQ_FREE.

Syntax **SEQ_ALLOC** *nb%, sizetype%, idtype%, seq%*

Parameters

nb%	INT(4)	I	size of the sequence to allocate
sizetype%	INT(2)	I	size allocated for an entry in the sequence
idtype	INT(2)	I	structure type
seq%	POINTER	I/O	address of the sequence

Notes

1. You don't need to know the exact number of items you'll manipulate: you can reallocate memory for a sequence.
2. When you allocate a sequence, 32 additional items are reserved in memory to avoid excessive reallocation, which would take up too much processing time.
3. If the memory could not be allocated, the address of the sequence is set to zero (0). Otherwise, the memory just allocated is automatically initialized to NULL.
4. The *idtype* parameter designates a basic type using one of the following constants:

```
CONST SEQ_TYPE_INT%      0
CONST SEQ_TYPE_INTEGER%  0
CONST SEQ_TYPE_STRING%   1
CONST SEQ_TYPE_CSTRING%  2
CONST SEQ_TYPE_NUM%      3
CONST SEQ_TYPE_CHAR%     4
CONST SEQ_TYPE_SEGMENT%  6
CONST SEQ_TYPE_POINTER%  7
CONST SEQ_TYPE_SEQUENCE% 8
CONST SEQ_TYPE_DYNSTR%   9
```

5. The parameter *idtype* is mainly for the Remote functions under NatStar. Under NSDK, this constant does not have any use but is kept for compatibility reasons.
6. Remember to free your sequence correctly afterwards by means of a SEQ_FREE, otherwise the *Memory Map* debug window appears.

Example 1

```
local i%          ;Sequence counter
local POINTER seq
local POINTER ADDR%
```

```
; the typed pointers have to be used prefixed by @ (it's
; references)
SEQ_ALLOC 10, SIZEOF SEQ_PINT, SEQ_TYPE_INT%,SEQ
For i% = 0 to seq_GET_NB_ENTRY%(SEQ) - 1
    Addr% = seq_GET_ENTRY%(SEQ,i%)
    SEQ_PINT(addr%).I = i%
EndFor
for i% = 0 to seq_GET_NB_ENTRY%(SEQ) - 1
    addr% = SEQ_GET_ENTRY%(SEQ,i%)
    INSERT AT END "Seq de rang"&i%&&SEQ_PINT(addr%).I TO LB
endfor

SEQ_FREE SEQ
```

See also SEQ_FREE, SEQ_GET_ENTRY%, SEQ_GET_NB_ENTRY%, SEQ_SET_NB_ENTRY

SEQ_SET_NB_ENTRY instruction

Sets the number of elements in a sequence.

Syntax	**SEQ_SET_NB_ENTRY** *seq%, nb%*			
Parameters	*seq%*	POINTER	I	handle of the sequence
	nb%	INT(2)	I	sequence number to set

Example

```
; A sequence with the handle seq% is filled up to index 1000.
; Empty this sequence up to index 10
SEQ_SET_NB_ENTRY seq%, 10
; Data stored in the 990 withdrawn sequences will be kept in
; memory but will be inaccessible if one goes through the list
; while using SEQ_GET_NB_ENTRY%. Then if one redefines the
; sequence with 1000 elements, the 990 elements will be restored
; entirely.
```

See also SEQ_ALLOC, SEQ_FREE, SEQ_GET_ENTRY%, SEQ_GET_NB_ENTRY%

SEQ_GET_NB_ENTRY% function

Returns the total number of instances in a data structure.

Syntax	**SEQ_GET_NB_ENTRY%** *(seq)*			
Parameter	*seq*	POINTER	I	handle of the sequence
Return value	INT(2)			

Example

```
local i%          ;Sequence counter
local POINTER seq
local nb%         ;items numbers of the sequence
local POINTER hli ;Found Instance handle
; the typed pointers have to be used prefixed by @ (it's
; references)
SEQ_ALLOC 0, SizeOf SEQ_PDYNSTR, SEQ_TYPE_DYNSTR%, seq
nb%=10
; Ecriture
i% = 0
while i% < nb%
    hli = SEQ_GET_ENTRY%(seq, i%)
    if hli = 0
          ns_trace ">>> !!! hdl d'entrée nulle!"
          break
    endif
    SEQ_PDYNSTR(hli).S="Antoine Tawil the "&i%
    i% = i% + 1
endwhile

; Lecture
i% = 0
while i% < SEQ_GET_NB_ENTRY%(seq)-1
    hli = SEQ_GET_ENTRY%(seq, i%)
    if hli = 0
          ns_trace ">>> !!! hdl d'entrée nulle!"
          break
    endif
    ns_trace "Dynstr "&& SEQ_PDYNSTR(hli).S
    INSERT AT END "Dynstr "&& SEQ_PDYNSTR(hli).S TO LB
    i% = i% + 1
endwhile

SEQ_FREE Seq
```

See also SEQ_ALLOC, SEQ_FREE, SEQ_GET_ENTRY%, SEQ_SET_NB_ENTRY

SEQ_INSERT_ENTRY% function

Append an item in the sequence at the *index%* position.

Syntax	**SEQ_INSERT_ENTRY%** (*seq%, index%*)			
Parameters	*seq%*	POINTER	I	handle of the sequence
	index%	INT	I	index of the sequence

Notes

1. The sequence index starts at zero.

2. Returns a pointer on the inserted item. The returned pointer points on a memory region in conformity with the type defined at the time of the SEQ_ALLOC and initialized with NULL.

3. If *index*% position is less than *seq%* sequence size the sequence is extended to include the *index%* position.

4. If the *index%* position is higher than *seq%* sequence size, the sequence is increased of as much. The entries thus created are initialized with NULL.

5. For cases 3 and 4 above, the number of sequences returned by the function SEQ_GET_NB_ENTRY% will be in conformity with the index request.

SEQ_GET_ENTRY% function

Returns the pointer to a data structure instance, known as an 'entry'.

Syntax **SEQ_GET_ENTRY%** *(seq%, index%)*

Parameters

seq%	POINTER	I	handle of the sequence
index%	INT(2)	I	index of the data structure instance. The first index is 0.

Return value POINTER

Notes

1. If the requested instance index doesn't exist (i.e. hasn't yet been allocated), the sequence is automatically reallocated and a storage area of 32 items is added to it.

2. If the position *index%* is greater than the size of the sequence, the sequence is increased as much, thus the newly created entries will be initialized with NULL. The number of sequences returned by the function SEQ_GET_NB_ENTRY% will be in conformity with the index request.

Example

```
local i%          ;Sequence counter
local POINTER seq
local POINTER ADDR%
; the typed pointers have to be used prefixed by @ (it's
; references)
SEQ_ALLOC 10, SIZEOF SEQ_PINT, SEQ_TYPE_INT% , SEQ
For i% = 0 to seq_GET_NB_ENTRY%(SEQ) - 1
    Addr% = seq_GET_ENTRY%(SEQ,i%)
    SEQ_PINT(addr%).I = i%
EndFor
for i% = 0 to seq_GET_NB_ENTRY%(SEQ) - 1
    addr% = SEQ_GET_ENTRY%(SEQ,i%)
    INSERT AT END "Seq de rang"&i%&&SEQ_PINT(addr%).I TO LB
endfor

SEQ_FREE SEQ
```

See also SEQ_ALLOC, SEQ_FREE, SEQ_GET_NB_ENTRY%, SEQ_SET_NB_ENTRY

SEQ_FREE instruction

Frees all the memory occupied by a sequence.

Syntax **SEQ_FREE** *seq%*

Parameter *seq%* POINTER I handle of the sequence

Example

```
local i%          ;Sequence counter
local POINTER seq
local POINTER ADDR%
; the typed pointers have to be used prefixed by a @ (it's
; references)
SEQ_ALLOC 10, SIZEOF SEQ_PINT, SEQ_TYPE_INT% , SEQ
For i% = 0 to seq_GET_NB_ENTRY%(SEQ) - 1
    Addr% = seq_GET_ENTRY%(SEQ,i%)
    SEQ_PINT(addr%).I = i%
EndFor
for i% = 0 to seq_GET_NB_ENTRY%(SEQ) - 1
    addr% = SEQ_GET_ENTRY%(SEQ,i%)
    INSERT AT END "Seq de rang"&i%&&SEQ_PINT(addr%).I TO LB
endfor

SEQ_FREE SEQ
```

See also SEQ_ALLOC, SEQ_GET_ENTRY%, SEQ_GET_NB_ENTRY%, SEQ_SET_NB_ENTRY

SEQ_DELETE_ENTRY instruction

Deletes the *index%* position of the sequence.

Syntax **SEQ_DELETE_ENTRY** *seq%, index%*

Parameters	*seq%*	POINTER	I	handle of the sequence
	index%	INT	I	index of the sequence

Notes

1. The sequence index starts at zero.
2. If the position *index%* is higher than the size of the sequence, the sequence is increased of as much. The entries thus created are initialized with NULL.

Chapter 3 User Functions Library NSFCTUTIL

This chapter describes the library of NSFCTUTIL user functions that are supplied with the NS-DK tool.

This library is NS-DK specific.

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

You will find in this chapter

- The description and syntax of each of NSFCTUTIL library's components.
- How to define user functions.

Table of contents

Introduction

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).

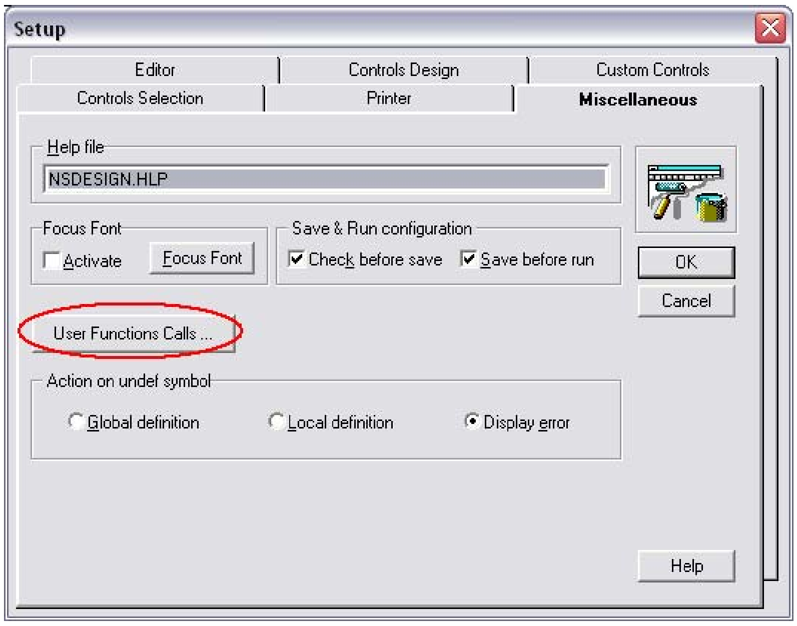
Prototypes of these functions as well as their names are imposed by Nat System. The name of the DLL in which these functions should be found will be configurable at the NS-Design tool level in the Setup dialogue box. Configuration should be carried out on each workstation (saved in the NSDKLOC.INI file).

Users do not need to encode all functions that can potentially be called. The tool manages the menus as well as calls of different functions based on whether they are in the user DLL or not.

The definition of constants and segments that are used for calling user functions are introduced in the NSFCTUTIL.NCL file.

Name of the user DLL

The detail window of these user functions is called from the *Setup* window (accessed from the *Option/Setup* menu), in the *Miscellaneous* tab.


Setup

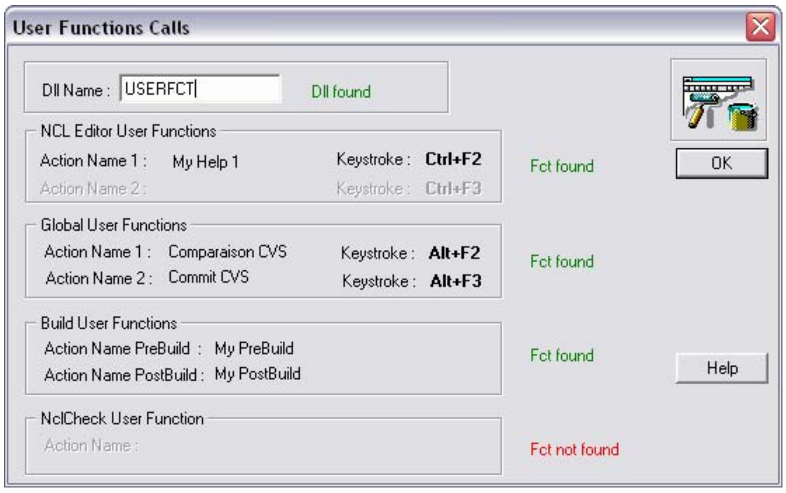
Help file: NSDESIGN.HLP

Focus Font: Activate ☐

Save & Run configuration: Check before save ☑, Save before run ☑

Action on undef symbol: Global definition ○, Local definition ○, Display error ●

Click the *User Functions Calls* button to see the following window:


User Functions Calls

Dll Name : USERFCT — Dll found

NCL Editor User Functions — Fct found

Action Name 1 :	My Help 1	Keystroke :	Ctrl+F2
Action Name 2 :		Keystroke :	Ctrl+F3

Global User Functions — Fct found

Action Name 1 :	Comparaison CVS	Keystroke :	Alt+F2
Action Name 2 :	Commit CVS	Keystroke :	Alt+F3

Build User Functions — Fct found

Action Name PreBuild :	My PreBuild
Action Name PostBuild :	My PostBuild

NclCheck User Function — Fct not found

Action Name :

The *Dll Name* field indicates the name of the user DLL in which user functions should be encoded.

☞ If there are consistency problems with the NS-DK tool, different keyboard shortcuts corresponding to the four user functions that can potentially be in the user DLL are imposed and cannot be modified.

In the *Ncl Editor User Functions* group, the *Action Name 1* and *Action Name 2* fields correspond to the text that will be displayed in the *Edit* menu when the focus is on the NCL editor.

In the *Global User Functions* group, the *Action Name 1* and *Action Name 2* fields correspond to the text of the help bubbles of the user functions' buttons.

In our example, the user chooses the name of their DLL (USERFCT.DLL) and then the functions present are displayed. We then note that the second function in the NCL editor was not encoded (*ActionName2* is greyed out) and neither is the user function that was called after confirming the code.

Depending on the context in which these functions are called, two data exchange segments should be sent to the following:

- SEG_FCTPARAMIN
- SEG_FCTPARAMOUT

Installation and use

After installing NS-DK, you will find NSFCTUTIL.NCL in the <NSDK>\NCL directory, where <NSDK> represents the installation directory.

You can consult the NSFCTUTIL.NCL file with NS-Design.

Functional categories of the NSFCTUTIL library

Here is the list of functionalities that allow us to manage user functions.

NSFCTUTIL library index

NS_USERFUNCT_QUERYACTIONNAME$ function

This function allows NS-Design to configure the text in the user menus (*ActionName1* etc.) at the tool level, as well as whether to call different functions or not.

Syntax	**NS_USERFUNCT_QUERYACTIONNAME$** (*NS_USRFCT_TYPE%*)		
Parameter	*NS_USRFCT_TYPE%*	INT	one of the *NS_USRFCT_*%* constants
Value returned	CSTRING		

Example

In our example, the second user function will not be called because the text returned is empty. The presence (or lack) of different user functions at the level of the detail window of user controls can be viewed.

```
Function NS_USERFUNCT_QUERYACTIONNAME$(Int NS_USRFCT_TYPE%)
Return Cstring

Local Cstring  ActionName$

ActionName$ = ""

Evaluate NS_USRFCT_TYPE%

    Where   NS_USRFCT_NCLCHECK%
                 ActionName$ = "Check"
    EndWhere
    Where NS_USRFCT_PREBUILD%
                 ActionName$ = "My PreBuild"
    EndWhere
    Where NS_USRFCT_POSTBUILD%
                 ActionName$ = "My PostBuild"
    EndWhere
    Where NS_USRFCT_NCLEDITOR1%
                 ActionName$ = "My Help 1"
    EndWhere
    Where NS_USRFCT_NCLEDITOR2%
                 ActionName$ = ""
    EndWhere
    Where NS_USRFCT_GLOBAL1%
                 ActionName$ = "Comparison CVS"
    EndWhere
    Where NS_USRFCT_GLOBAL2%
                 ActionName$ = "Commit CVS"
    EndWhere
    else
          ActionName$ = ""

EndEvaluate
```

```
Return ActionName$
EndFunction
```

See also NS_USRFCT_*%

NS_USERFUNCT_NCLCHECK% function

This function is called (if it exists) when the NCL code is being confirmed ([F8] key).

Syntax	**NS_USERFUNCT_NCLCHECK%** (*NsParamIn*, *NsParamOut*)			
Parameters	*NsParamIn*	SEGMENT	O	SEG_FCTPARAMIN segment
	NsParamOut	SEGMENT	O	SEG_FCTPARAMOUT segment

Value returned INT

TRUE%, or FALSE% if an error has been found in the code.

Comments

1. When this function is called, all of the segment's fields are filled in (except for the compilation configuration)

2. If an error has been found in the code, the *Err_Message$*, *Col_Number%* and *Line_Number%* fields of the SEG_FCTPARAMOUT segment should be filled in. The error message will then be displayed in the status bar of the NCL editor and the cursor will be at the coordinates supplied by the two fields *Col_Number%* and *Line_Number%*.

3. The NS_USERFUNCT_NCLCHECK% user function will not be called after pressing the [F8] key if it has not detected a syntax error or when saving (if the *Check before Save* option is selected in the *Setup* dialogue box in the *Miscellaneous* tab)

4. Allocating and removing allocation of segments that have been sent as parameters is carried out by NS-Design.

See also SEG_FCTPARAMIN, SEG_FCTPARAMOUT

NS_USERFUNCT_BUILD% function

This function allows us to add control functions to the project before and/or after constructing it (*Build*).

Syntax **NS_USERFUNCT_BUILD%** (*BuildPos%*, *NsParamIn*, *ppointer*)

Parameters

BuildPos%	INT	I	integer that can take the NS_USRFCT_PREBUILD% or NS_USRFCT_POSTBUILD% value
NsParamIn	SEGMENT	O	SEG_FCTPARAMIN segment
ppointer	SEGMENT	O	segment typed on a sequence that contains SEG_FCTPARAMOUT type segments

Value returned INT

TRUE%, the compilation is carried out as normal.

Comments

1. When this is called, only fields that contain the name of the project as well as the configuration of the current compilation will be filled in.
2. If the NS_USERFUNCT_BUILD% function (with the *BuildPos%* = NS_USRFCT_PREBUILD% parameter) returns FALSE%, the compilation will not be carried out and the *Log* window will appear on top and errors that were filled in at the output sequence will be displayed.
3. If the NS_USERFUNCT_BUILD% function (with the *BuildPos*% = NS_USRFCT_POSTBUILD% parameter) returns FALSE%, the *Log* window will appear on top and errors that were filled in at the output sequence will be displayed.
4. The third parameter is a pointer typed onto a sequence (linked list) that is allocated and unallocated by NS-Design. When the user function fills in SEG_FCTPARAMOUT type elements, the *Severity%* parameter should be filled in.
5. Allocating and removing allocation of different structures that have been sent as parameters is carried out by NS-Design.

Example

```
Function NS_USERFUNCT_BUILD% (Int BuildPos%,SEG_FCTPARAMIN
@NSPARAMIN, SEQ_PPOINTER @ppointer) return int

    Local SEG_FCTPARAMOUT @NSPARAMOUT

           Evaluate BuildPos%

                        Where NS_USRFCT_PREBUILD%

                               Message NSPARAMIN.Proj_Name$
,NSPARAMIN.Config_Name$

                               @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                               NSPARAMOUT.Err_Message$ = "Error
before 1"
                               NSPARAMOUT.Severity% = 0
                               NSPARAMOUT.Col_Number% = -1
                               NSPARAMOUT.Line_Number% = -1

                               @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                               NSPARAMOUT.Err_Message$ = "Error
before 2"
                               NSPARAMOUT.Severity% =
NSWARNINGERROR%
                               NSPARAMOUT.Col_Number% = -1
                               NSPARAMOUT.Line_Number% = -1

                               @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                               NSPARAMOUT.Err_Message$ = "Error
before 3"
                               NSPARAMOUT.Severity% =
NSFATALERROR%
                               NSPARAMOUT.Col_Number% = -1
                               NSPARAMOUT.Line_Number% = -1
                               Return FALSE%


                        EndWhere

                        Where NS_USRFCT_POSTBUILD%

                           Message    NSPARAMIN.Proj_Name$
,NSPARAMIN.Config_Name$

                               @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                               NSPARAMOUT.Err_Message$ = "Error
after 1"
                               NSPARAMOUT.Severity% =
NSWARNINGERROR%
                               NSPARAMOUT.Col_Number% = -1
                               NSPARAMOUT.Line_Number% = -1

                               @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
```

```
                              NSPARAMOUT.Err_Message$ = "Error
after 2"
                              NSPARAMOUT.Severity% =
NSFATALERROR%
                              NSPARAMOUT.Col_Number% = -1
                              NSPARAMOUT.Line_Number% = -1

                              @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                              NSPARAMOUT.Err_Message$ = "Error
after 3"
                              NSPARAMOUT.Severity% = 0
                              NSPARAMOUT.Col_Number% = -1
                              NSPARAMOUT.Line_Number% = -1
                               Return False%
                       EndWhere
                       ELSE
                       Return False%


                 EndEvaluate


    Return True%

EndFunction
```

See also Error management constants, NS_USRFCT_POSTBUILD%, NS_USRFCT_PREBUILD%, SEG_FCTPARAMIN, SEG_FCTPARAMOUT

NS_USERFUNCT_LOADPROJECT% Function

This user function will be called during the load of the project. A parameter indicates if the project is converted from PRJ to XNP.

The prototype of the called function is the following.

Syntax **NS_USERFUNCT_LOADPROJECT%** (*prj*, *prjconv*, *ppointer*)

Parameters

prj	SEG_XPRJ	I	SEG_XPRJ@ segment
prjconv	INT	I	true% or false%. Indicates if the project has been converted (from PRJ to XNP)
ppointer	SEGMENT	O	typed segment on a sequence containing segments of the SEG_FCTPARAMOUT type

Return value INT
TRUE% or FALSE%

Notes

1. The parameter *prj* is a specific segment allowing you to call the XPRJ_* functions of handling project.
2. *prjconv* is at true% if the project were converted from PRJ to XNP, false% otherwise.
3. The parameter *ppointer* is a typed pointer on a sequence (chained list) which is allocated and deallocated by NS-Design. When the user function enter the elements of the SEG_FCTPARAMOUT type, the parameter *Severity%* must be entered. The possible values are NSINFOERROR% (in this case the wording of the error will be preceded by `---`), NSWARNINGERROR% (in this case the wording of the error will be preceded by a yellow icon of warning), or NSFATALERROR% (in this case the wording of the error will be preceded by a red icon of warning).

Example

```
; FUNCTION called after loading any project,
; prjconv indicates if project was convert from PRJ to XNP
;----------------------------------------------------------------

Function NS_USERFUNCT_LOADPROJECT% (SEG_XPRJ @nsprj, int
prjconv, SEQ_PPOINTER @PPointer) return int
```

```
    local Cstring val
    local SEG_XPRJ @pGenConfig
    local SEG_XPrjErr err
    local ret%, count%, i%
    local configName$

    addToLog (PPointer,"Project loaded in user function",
NS_USRFCT__LEVEL_INFO%, -1, -1)

    if (prjconv) ; first convertion from PRJ file project ?
           addToLog (PPointer,"first convertion from PRJ",
NS_USRFCT__LEVEL_INFO%, -1, -1)
    endif

    ; number of generation config
    ret% = XPRJ_GetIntAttr% (nsprj, xpaGenCount%, count%)

    for i%=0 to count%-1

           ; retrieve generation config name
           ret% = XPRJ_GetTextAttrIdx% (nsprj, xpaCfgName%, i%,
configName$)
           addToLog (PPointer,"build configuration name : " &
configName$, NS_USRFCT__LEVEL_INFO%, -1, -1)

           ; retrieve child project pointer
           @pGenConfig = XPRJ_LoadConfig (nsprj, ConfigName$,
err)

           ; retrieving generation directories
           ret% = XPRJ_GetTextAttr% (pGenConfig,
xpaCfgSourcesPath%, val)
           addToLog (PPointer,"    directory for generated .C
files = " & val, NS_USRFCT__LEVEL_INFO%, -1, -1)
           ret% = XPRJ_GetTextAttr% (pGenConfig,
xpaCfgBinariesPath%, val)
           addToLog (PPointer,"    directory for generated .EXE
files = " & val, NS_USRFCT__LEVEL_INFO%, -1, -1)
           ret% = XPRJ_GetTextAttr% (pGenConfig,
xpaCfgDLLsPath%, val)
           addToLog (PPointer,"    directory for generated .DLL
files = " & val, NS_USRFCT__LEVEL_INFO%, -1, -1)

    endfor

EndFunction ; NS_USERFUNCT_LOADPROJECT%
```

See also NS_USRFCT__GLOBAL1%, NS_USRFCT__GLOBAL2%, SEG_FCTPARAMIN, SEG_FCTPARAMOUT

NS_USERFUNCT_NCLEDITOR% function

Two user functions can be run from the NCL editor. These functions will appear at the *Edit* menu level if they are in the user DLL. They can be called by the keyboard shortcut [Ctrl]+[F2] and [Ctrl]+[F3]. If there are any consistency problems with the NS-Design tool, these keyboard shortcuts are imposed by Nat System and cannot be modified.

The function called should be defined according to the following prototype.

Syntax **NS_USERFUNCT_NCLEDITOR%** (*FctNum%, NsParamIn, ParamOut)*

Parameters	*FctNum%*	INT	I	Integer that identifies the two user functions, which can be called
	NsParamIn	SEGMENT	O	segment typed on input parameters entered by NSDESIGN (SEG_FCTPARAMIN)
	ParamOut	SEGMENT	O	Segment typed on output parameters entered by the user DLL (SEG_FCTPARAMOUT)

Value returned INT

TRUE% or FALSE%

Comments

1. The *FctNum%* parameter allows us to identify which user function, 1 or 2, was called (respectively identified by the constants NS_USRFCT_NCLEDITOR1% and NS_USRFCT_NCLEDITOR2%)
2. When this function is called, all of the fields are filled in except for the compilation configuration one.
3. When returned, these two user functions should have the return code of the function set to TRUE% if everything is OK and to FALSE% if an error in the code was found. In this case, the *Err_Message*$, *Col_Number*% and *Line_Number*% fields of the SEG_FCTPARAMOUT segment should be filled in. The error message will then be displayed in the status bar of the NCL editor and the cursor will be at the coordinates supplied by the two fields *Col_Number*% and *Line_Number*%.
4. Allocating and removing allocation of different structures that have been sent as parameters is carried out by NS-Design.

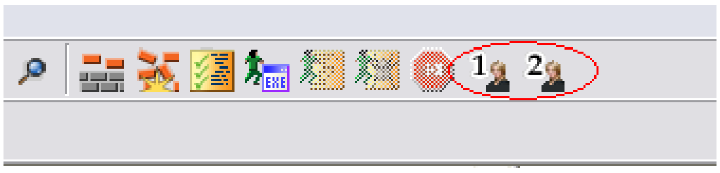
Example

```
Function NS_USERFUNCT_NCLEDITOR% (Int FctNum%,SEG_FCTPARAMIN
@NSPARAMIN, SEG_FCTPARAMOUT  @NSPARAMOUT) Return Int
    Local pointer hvisu%
    Local Int ret%(2)

    Evaluate FctNum%
          Where NS_USRFCT_NCLEDITOR1%
                       …
          EndWhere
          Where NS_USRFCT_NCLEDITOR2%
                       …
          EndWhere
    EndEvaluate

Return ret%
EndFunction
```

See also NS_USRFCT_NCLEDITOR1%, NS_USRFCT_NCLEDITOR2%, SEG_FCTPARAMIN, SEG_FCTPARAMOUT

NS_USERFUNCT_GLOBAL% function

Two user functions can be run, no matter what windows are open under NS-Design. These functions will appear at the *Build* menu level (if they are present in the user DLL) as well depending on whether two extra buttons are present in NS-Design's toolbar.



They can be called by the keyboard shortcut [Alt]+[F2] and [Alt]+[F3]. If there are any consistency problems with the NS-Design tool, these keyboard shortcuts are imposed by Nat System and cannot be modified.

The prototype of the function called is as follows.

Syntax **NS_USERFUNCT_GLOBAL%** (*FctNum%*, *NsParamIn*, *ppointer*)

Parameters	*FctNum%*	INT	I	Integer that identifies the two user functions, which can be called
	NsParamIn	SEGMENT	O	segment typed on input parameters entered by NS-Design (SEG_FCTPARAMIN)
	ppointer	SEGMENT	O	segment typed on a sequence that contains SEG_FCTPARAMOUT type segments

Value returned INT

TRUE% or FALSE%

Comments

1. The *FctNum*% parameter allows us to identify which user function, 1 or 2, was called (respectively identified by the constants NS_USRFCT_GLOBAL1% and NS_USRFCT_GLOBAL2%)

2. When this is called, only the field that contains the name of the project will be filled in.

3. If the function called returns FALSE%, the *Log* window will appear on top and errors that were filled in at the output sequence will be displayed.

4. The *ppointer* parameter is a pointer typed onto a sequence (linked list) that is allocated and unallocated by NS-Design. When the user function fills in SEG_FCTPARAMOUT type elements, the *Severity*% parameter should be filled in. The values that are possible are: NSINFOERROR% (in this case, the text of the error will be preceded by '---', NSWARNINGERROR% (in this case, the text of the error will be preceded by a yellow warning icon), or NSFATALERROR% (in this case the text of the error will be preceded by a red warning icon).

5. Allocating and removing allocation of different structures that have been sent as parameters is carried out by NS-Design.

Example

```
Function NS_USERFUNCT_GLOBAL% (Int FctNum%,SEG_FCTPARAMIN
@NSPARAMIN, SEQ_PPOINTER @PPointer) return int
    Local pointer hvisu%
    Local Int ret%(2)
    Local SEG_FCTPARAMOUT @NSPARAMOUT



                 Evaluate FctNum%
                        Where NS_USRFCT_GLOBAL1%
                               Message   "Ctrl Glob1",
NSPARAMIN.Proj_Name$

                               @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                               NSPARAMOUT.Err_Message$ = "Error
Glob 1 1"
                               NSPARAMOUT.Severity% =
NSWARNINGERROR%
                               NSPARAMOUT.Col_Number% = -1
                               NSPARAMOUT.Line_Number% = -1

                               @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                               NSPARAMOUT.Err_Message$ =
"ErrorGlob 1 2"
                               NSPARAMOUT.Severity% =
NSFATALERROR%
                               NSPARAMOUT.Col_Number% = -1
                               NSPARAMOUT.Line_Number% = -1

                               @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                               NSPARAMOUT.Err_Message$ = "Error
Glob 1 3"
                               NSPARAMOUT.Severity% =
NSWARNINGERROR%
                               NSPARAMOUT.Col_Number% = -1
                               NSPARAMOUT.Line_Number% = -1
                        EndWhere
                        Where NS_USRFCT_GLOBAL2%
```

```
                              Message   "Ctrl Glob2",
NSPARAMIN.Proj_Name$

                              @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                              NSPARAMOUT.Err_Message$ = "Error
Glob 2 1"
                              NSPARAMOUT.Severity% = 0
                              NSPARAMOUT.Col_Number% = -1
                              NSPARAMOUT.Line_Number% = 0

                              @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                              NSPARAMOUT.Err_Message$ =
"ErrorGlob 2 2"
                              NSPARAMOUT.Severity% =
NSFATALERROR%
                              NSPARAMOUT.Col_Number% = -1
                              NSPARAMOUT.Line_Number% = -1

                              @NSPARAMOUT =
SEG_FCTPARAMOUT(seq_INSERT_ENTRY%(@PPointer,0))
                              NSPARAMOUT.Err_Message$ = "Error
Glob 2 3"
                              NSPARAMOUT.Severity% =
NSWARNINGERROR%
                              NSPARAMOUT.Col_Number% = -1
                              NSPARAMOUT.Line_Number% = -1
                        EndWhere
                 EndEvaluate


Return ret%
EndFunction
```

See also NS_USRFCT_GLOBAL1%, NS_USRFCT_GLOBAL2%, SEG_FCTPARAMin, SEG_FCTPARAMOUT

Segment SEG_FCTPARAMIN

Segment used by user functions for input data entered by NS-Design.

Syntax

```
Segment SEG_FCTPARAMIN
    Ncl_Fic$
    Proj_Name$
    Config_Name
    Selected_topic$
    LibNcl_Name
    Scr_Name$
    Ctrl_Name$
    Event_Name$
    Target_Name$
EndSegment
```

Fields

Ncl_Fic$	CSTRING	temporary file that contains the NCL code
Proj_Name$	CSTRING	project name (.xnp)
Config_Name	CSTRING	name of the compilation configuration
Selected_topic$	CSTRING	selected topic
LibNcl_Name	CSTRING	name of the NCL library
Scr_Name$	CSTRING	screen name
Ctrl_Name$	CSTRING	name of the control
Event_Name$	CSTRING	event name
Target_Name$	CSTRING	target name

Comments

1. Depending on the context, all fields cannot be filled in.
2. Their internal declaration is:

```
Segment SEG_FCTPARAMIN
      Cstring Ncl_Fic$
      Cstring Proj_Name$
      Cstring Config_Name$
      Cstring Selected_topic$
      Cstring LibNcl_Name$
      Cstring Scr_Name$
      Cstring Ctrl_Name$
      Cstring Event_Name$
      Cstring Target_Name$
EndSegment
```

See also SEG_FCTPARAMOUT

Segment SEG_FCTPARAMOUT

Segment used by user functions for output data entered by the user.

Syntax

Segment SEG_FCTPARAMOUT

Err_Message$

Col_Number%

Line_Number%

Severity%

EndSegment

Fields

Err_Message$	CSTRING	text of the error message
Col_Number%	INT	column number where the error is located
Line_Number%	INT	line number where the error is located
Severity%	INT	severity level of the error

Comments

1. Depending on the context, all fields cannot be filled in.
2. The *Severity%* field contains one of the NSINFOERROR%, NSWARNINGERROR% or NSFATALERROR% constants.
3. Their internal declaration is:

```
Segment SEG_FCTPARAMIN
      Cstring Err_Message$
      Int Col_Number%
      Int Line_Number%
      Int Severity%
EndSegment
```

See also

SEG_FCTPARAMIN, NSINFOERROR%, NSWARNINGERROR%, NSFATALERROR%

NS_USRFCT_*% constants

The NS_USRFCT_*% constants correspond to the identifier of different user functions that were potentially created by the user.

Syntax **NS_USRFCT_NCLCHECK%**

NS_USRFCT_PREBUILD%

NS_USRFCT_POSTBUILD%

NS_USRFCT_NCLEDITOR1%

NS_USRFCT_NCLEDITOR2%

NS_USRFCT_GLOBAL1%

NS_USRFCT_GLOBAL2%

Comments

1. These constants are to be used most notably as parameters in the NS_USRFCT_QUERYACTIONNAME$ function.
2. They can be explained as follows:

NS_USRFCT_NCLCHECK%	Identifies the user function that was run after confirming the NCL code ([F8] key)
NS_USRFCT_PREBUILD%	Identifies the user function that was run before the Build.
NS_USRFCT_POSTBUILD%	Identifies the user function that was run after the Build.
NS_USRFCT_NCLEDITOR1%	Identifies the user function that was run when editing an NCL using the CTRL+F2 shortcut.
NS_USRFCT_NCLEDITOR2%	Identifies the user function that was run when editing an NCL using the CTRL+F3 shortcut.
NS_USRFCT_GLOBAL1%	Identifies the user function that was run using the ALT+F2 shortcut.
NS_USRFCT_GLOBAL2%	Identifies the user function that

was run using the ALT+F2 shortcut.

3. Their internal declaration is:

```
Const NS_USRFCT_NCLCHECK%    1
Const NS_USRFCT_PREBUILD%    2
Const NS_USRFCT_POSTBUILD%   3
Const NS_USRFCT_NCLEDITOR1%  4
Const NS_USRFCT_NCLEDITOR2%  5
Const NS_USRFCT_GLOBAL1%     6
Const NS_USRFCT_GLOBAL2%     7
```

See also NS_USRFCT_QUERYACTIONNAME$

Error management constants

The error management constants allow us to specify the severity level of the error in the SEG_FCTPARAMOUT segment.

Syntax **NSINFOERROR%**
NSWARNINGERROR%
NSFATALERROR%

Comments

1. They can be explained as follows:

NSINFOERROR%	Information
NSWARNINGERROR%	Warning error
NSFATALERROR%	Fatal error

2. Their internal declaration is:

```
Const NSINFOERROR%    0
Const NSWARNINGERROR% 1
Const NSFATALERROR%   2
```

3. When displaying errors in the *Log* window, if the NSINFOERROR% constant is set, the text of the error will then be preceded by "---". If the constant is NSWARNINGERROR%, the text of the error will then be preceded by a yellow warning icon. And finally, if NSFATALERROR% is set, the text of the error will then be preceded by a red warning icon.

See also SEG_FCTPARAMOUT