XML SAX Library

This chapter introduces the XML SAX library integrated to the Nat System development tools.

In this chapter you will find

- The components of this library, classified in function categories
- The reference of this library's components

Table of contents

Installation

Declare NsSaxXml.NCL in the libraries that are necessary in developing your application.

Ensure that the NWXML.DLL and NWXMLLIB files have been placed in one of the PATH directories under Windows.

NsSaxXml.NCL file

The verbs in this library (NsSaxXml) are described below. They are declared in a text file, written in NCL, called NsSaxXml.NCL. This file may also contain some complementary verbs (non-public API) which are not documented, so you can, if you wish, look at the file in order to have a complete reference for the library.

To look at the NsSaxXml.NCL file:

1. Navigate to the <NATSTAR>\NCL or <NSDK>\NCL or <NATWEB>\NCL directory.

   NB: <NATSTAR> represents NatStar's installation folder. The procedure is identical for NatWeb and NS-DK.

2. Open the NsSaxXml.NCL file with any text editor to see its contents.

Function categories of the NsSaxXml library

Here is the list of instructions and functions of the NsSaxXml library, organised by categories.

Starting and ending library use

Parser management

Analysis methods of node attributes

Methods to be used with a LOCATOR

LOCATOR	The **LOCATOR** is used to locate the cursor during flow processing. Thus, for example, the row and column number being analysed can be known.

Methods to be used with an exception handle

Typical examples of callback methods

The syntax analyser automatically calls on a method when an event is detected in the XML file. Seven events are detected by the library: start element detection (SAXH_STARTELEMENT), end element detection (SAXH_ENDELEMENT), data between two elements (SAXH_CHARACTERS), start processing an XML document (SAXH_STARTDOCUMENT), end processing (SAXH_ENDDOCUMENT), detecting a comment (SAXH_COMMENT), detecting a processing instruction.

These functions can be used by saving them in the XML syntax analyser with the NSAX_INIFNT instruction.

Error management

Reference of the NsSaxXml library

NSAX_INITIALIZE% function

This initialises a connection with the given session.

Syntax **NSAX_INITIALIZE%(***UNREPFLAGS%*,*EXPANDNAMESSPACES%***)**

Parameters

UNREPFLAGS%	INT(4)	I	flag
EXPANDNAMESSPACES%	INT(1)	I	expand/do not expand name spaces

Comments

1. The *UNREPFLAGS%* parameter indicates to the parser what must be done when it cannot represent a character in the character set.
   - cSAX_CharRef% displays the character as a reference.
   - cSAX_fail% failed operation
   - cSAX_replace% replaces the character with a replacement character.
2. These constants are declared as follows:

```
cSAX_CharRef%  0
cSAX_fail%     1
cSAX_replace%  2
```

3. The *EXPANDNAMESSPACES%* parameter indicates if the namespace's aliases should be turned off with their URIs. If this option is selected, the processing name spaces must be confirmed via the NSAX_PARSERSETFEATURE instruction with the cSAX_fgSAX2CoreNameSpaces option.

See also cSAX_* constants, NSAX_TERMINATE, NSAX_PARSERSETFEATURE

NSAX_TERMINATE instruction

This frees the resources allocated by the SAX library.

Syntax **NSAX_TERMINATE**

See also NSAX_INITIALIZE

NSAX_CREATEPARSER% function

This creates a “reader” object associated to the analyser in order to enable the analysis of a document.

Syntax	**NSAX_CREATEPARSER (***PDATA, LOCATION***)**			
Parameters	*PDATA*	POINTER	I	pointer supplied by the caller
	LOCATION	POINTER	I/O	returns a pointer in *LOCATION*
Value returned	INT			

Comments

1. The *PDATA* pointer is supplied by the caller and will be saved in the object created and then transmitted in each of the callback methods. Thus, the caller can associate information to the parser.
2. The LOCATION parameter returns a pointer into itself, which can be used during document analysis with the following functions:
   - NSAX_GETLINENUMBER%
   - NSAX_GETLINENUMBER%
   - NSAX_GETPUBLICID
   - NSAX_GETSYSTEMID

Example

```
Local PDATA%

INITIALIZE_PARSER Selection%(CB_CHARS),CHK_EXPANDNAMESSPACES=CHECKED%

New S_PRIVDATA,PDATA%


Insert At End "*** Create Parser result: " &
NSAX_CREATEPARSER(PDATA%,P_LOC) & " ***" TO LERR

S_PRIVDATA(PDATA%).LOCATOR = P_LOC
NSAX_PARSERSETFEATURE cSAX_fgSAX2CoreValidation, \
CHK_COREVALIDATION=CHECKED%
;Check box CHK_COREVALIDATION checked

Insert At End "*** Parse result : " & NSAX_STARTPARSER(CB_XMLFILE) & "
***" TO LERR

NSAX_RELEASEPARSER
Dispose PDATA%
```

```
NSAX_TERMINATE
```

See also NSAX_RELEASEPARSER**,** NSAX_PARSERSETFEATURE, NSAX_STARTPARSER

NSAX_RELEASEPARSER instruction

This releases the object used for document analysis.

Syntax **NSAX_RELEASEPARSER**

Example

```
Local PDATA%

INITIALIZE_PARSER Selection%(CB_CHARS),CHK_EXPANDNAMESSPACES=CHECKED%

New S_PRIVDATA,PDATA%


Insert At End "*** Create Parser result: " &
NSAX_CREATEPARSER(PDATA%,P_LOC) & " ***" TO LERR

S_PRIVDATA(PDATA%).LOCATOR = P_LOC
NSAX_PARSERSETFEATURE cSAX_fgSAX2CoreValidation, \
CHK_COREVALIDATION=CHECKED%
;Check box CHK_COREVALIDATION checked

Insert At End "*** Parse result : " & NSAX_STARTPARSER(CB_XMLFILE) & "
***" TO LERR

NSAX_RELEASEPARSER
Dispose PDATA%

NSAX_TERMINATE
```

See also NSAX_CREATEPARSER, NSAX_PARSERSETFEATURE, NSAX_STARTPARSER

NSAX_PARSERSETFEATURE instruction

This sets different behaviours of the syntax analyser, which were first created by NSAX_CREATEPARSER,

Syntax **NSAX_PARSERSETFEATURE (***FEATURE%, VAL%***)**

Parameters				
	FEATURE%	INT(4)	I	a cSAX*% constant
	VAL%	INT(4)	I	TRUE% or FALSE%

Example

```
Local PDATA%

INITIALIZE_PARSER Selection%(CB_CHARS),CHK_EXPANDNAMESSPACES=CHECKED%

New S_PRIVDATA,PDATA%


Insert At End "*** Create Parser result: " &
NSAX_CREATEPARSER(PDATA%,P_LOC) & " ***" TO LERR

S_PRIVDATA(PDATA%).LOCATOR = P_LOC
NSAX_PARSERSETFEATURE cSAX_fgSAX2CoreValidation, \
CHK_COREVALIDATION=CHECKED%
;Check box CHK_COREVALIDATION checked

Insert At End "*** Parse result : " & NSAX_STARTPARSER(CB_XMLFILE) & "
***" TO LERR

NSAX_RELEASEPARSER
Dispose PDATA%

NSAX_TERMINATE
```

See also cSAX_*% constants, NSAX_CREATEPARSER**,** NSAX_RELEASEPARSER, NSAX_STARTPARSER

cSAX_* constants

Behaviours of the syntax analyser.

Syntax **cSAX_fgSAX2CoreValidation**

cSAX_fgXercesDynamic

cSAX_fgSAX2CoreNameSpaces

cSAX_fgXercesSchema

cSAX_fgXercesSchemaFullChecking

cSAX_fgSAX2CoreNameSpacePrefixes

Comments

1. The constants are defined as follows:
   - cSAX_fgSAX2CoreValidation
     When set to *true*, the document should specify a validation grammar.
     By default: TRUE
   - cSAX_fgXercesDynamic
     When set to *true*, the analyser will validate the document only if a grammar has been supplied.
     By default: FALSE
   - cSAX_fgSAX2CoreNameSpaces
     When set to *true*, the document should contain a grammar that supports name spaces.
     By default: TRUE
   - cSAX_fgXercesSchema
     Validation of processing schemas. If set to *true*, the cSAX_fgSAX2CoreNameSpaces constant should be the same.
     By default: TRUE
   - cSAX_fgXercesSchemaFullChecking
     This examines the schema grammar in order to detect additional errors that are lengthy or that use up a lot of memory. It does not affect the verification level carried out on document instances that use grammar schemes. By default: FALSE
   - cSAX_fgSAX2CoreNameSpacePrefixes
     Set to *true*. This reports the names and original prefixed attributes used for namespace declarations.

By default: FALSE

2. The cSAX_CharRef%, cSAX_replace% and cSAX_fail% constants are used with the NSAX_INITIALIZE% function.
3. They are declared as follows:

```
cSAX_CharRef%  0
cSAX_fail%     1
cSAX_replace%  2
cSAX_fgSAX2CoreNameSpaces          103
cSAX_fgXercesSchema                104
cSAX_fgXercesSchemaFullChecking    105
cSAX_fgSAX2CoreNameSpacePrefixes   106
```

NSAX_STARTPARSER function

This starts the parameter specified document analysis.

Syntax **NSAX_STARTPARSER (***xmlfile$***)**

Parameter *xmlfile$* CSTRING I XML file name

Value returned INT

Example

```
Local PDATA%

INITIALIZE_PARSER Selection%(CB_CHARS),CHK_EXPANDNAMESSPACES=CHECKED%

New S_PRIVDATA,PDATA%


Insert At End "*** Create Parser result: " &
NSAX_CREATEPARSER(PDATA%,P_LOC) & " ***" TO LERR

S_PRIVDATA(PDATA%).LOCATOR = P_LOC
NSAX_PARSERSETFEATURE cSAX_fgSAX2CoreValidation, \
CHK_COREVALIDATION=CHECKED%
;Check box CHK_COREVALIDATION checked

Insert At End "*** Parse result : " & NSAX_STARTPARSER(CB_XMLFILE) & "
***" TO LERR

NSAX_RELEASEPARSER
Dispose PDATA%

NSAX_TERMINATE
```

See also NSAX_CREATEPARSER**,** NSAX_RELEASEPARSER, NSAX_PARSERSETFEATURE

NSAX_ATTRIBUTES_GETLENGTH% function

Returns the attribute number.

Syntax	**NSAX_ATTRIBUTES_GETLENGTH%** (*P*)			
Parameter	*P*	POINTER	I/O	pointer on an attribute
See also	NSAX_ATTRIBUTES_GETNAME%, NSAX_ATTRIBUTES_GETTYPE%, NSAX_ATTRIBUTES_GETVALUE%			

NSAX_ATTRIBUTES_GETNAME% function

Returns the attribute name in the form of a pointer on a DynStr.

Syntax **NSAX_ATTRIBUTES_GETNAME%** (*P, INDEX, EXPANDNAMESSPACES%*)

Parameters				
	P	DYNSTR	I/O	Returns the name in the form of a pointer on a DynStr. This pointer is a DynStr. If it is different to NULL, it should be then freed with NSAX_DISPOSE.
	INDEX	INT	I	attribute index
	EXPANDNAMESSPACES%	INT(1)	I	flag that indicates whether namespaces are extended or not with their URIs

See also NSAX_DISPOSE, NSAX_ATTRIBUTES_GETLENGTH%, NSAX_ATTRIBUTES_GETTYPE%, NSAX_ATTRIBUTES_GETVALUE%

NSAX_ATTRIBUTES_GETTYPE% function

Returns the attribute name in the form of a pointer on a DynStr.

Syntax **NSAX_ATTRIBUTES_GETTYPE% (***P*, *INDEX***)**

Parameters				
	P	DYNSTR	I/O	Returns the type in the form of a pointer on a DynStr. This pointer is a DynStr. If it is different to NULL, it should be then freed with NSAX_DISPOSE.
	INDEX	INT	I	attribute index

Comment

1. The possible types are:

"NOTATION" A series of notations separated by a vertical line.

Each one should exactly correspond to a name declared in the DTD.

```
NOTATION (val1 | val2 | ..)
```

"CDATA" Accepts any string of characters.

```
<!ATTLIST elmt_name attrib_name CDATA>
```

"ENTITY" Defines a non-parsed entity declared in a DTD as the attribute value.

```
<!ATTLIST elmt_name attrib_name ENTITY
#option>
```

"ENTITIES" Defines several non-parsed entities (separated by spaces) declared in a DTD as the attribute value.

```
<!ATTLIST elmt_name attrib_name ENTITIES
#option>
```

"ID" Indicates that the attribute has a unique value for each element.

The value can contain one or several letters, numbers, dots (.), hyphens (-), or underscores (_) and a colon (:).

```
<!ATTLIST elmt_name attrib_name ID #option>
```

"IDREF" Indicates that the attribute's value refers to an ID of another attribute.

```
<!ATTLIST elmt_name attrib_name IDREF
#option>
```

"IDREFS"	Identical to IDREF except that the attribute's value can refer to several IDs, each value being separated by a space. `<!ATTLIST elmt_name attrib_name IDREFS #option>`
"NMTOKEN"	Indicates that the attribute's value is a string of characters which can contain one or several letters, numbers, dots (.), hyphens (-), or underscores (_) and a colon (:). `<!ATTLIST elmt_name attrib_name NMTOKEN #option>`
"NMTOKENS"	Identical to NMTOKEN except that the attribute can have several values separated by spaces. `<!ATTLIST elmt_name attrib_name NMTOKENS #option>`

See also NSAX_DISPOSE, NSAX_ATTRIBUTES_GETLENGTH%, NSAX_ATTRIBUTES_GETNAME%, NSAX_ATTRIBUTES_GETVALUE%

NSAX_ATTRIBUTES_GETVALUE% function

Returns the possible value of an attribute in the form of a pointer on a DynStr.

Syntax **NSAX_ATTRIBUTES_GETVALUE% (***P, INDEX***)**

Parameters				
	P	DYNSTR	I/O	Returns the value in the form of a pointer on a DynStr. This pointer is a DynStr. If it is different to NULL, it should be then freed with NSAX_DISPOSE.
	INDEX	INT	I	attribute index

See also NSAX_DISPOSE, NSAX_ATTRIBUTES_GETLENGTH%, NSAX_ATTRIBUTES_GETNAME%, NSAX_ATTRIBUTES_GETTYPE%

NSAX_GETCOLUMNNUMBER% function

Returns the column number

Syntax	**NSAX_GETCOLUMNNUMBER% (*P*)**			
Parameter	*P*	POINTER	I	Pointer on a "Locator"
See also	NSAX_GETLINENUMBER%, NSAX_GETPUBLICID, NSAX_GETSYSTEMID			

NSAX_GETLINENUMBER% function

Returns the row number

Syntax	**NSAX_GETLINENUMBER% (*P*)**			
Parameter	*P*	POINTER	I	Pointer on a "Locator"
See also	NSAX_GETCOLUMNNUMBER%, NSAX_GETPUBLICID, NSAX_GETSYSTEMID			

NSAX_GETPUBLICID function

Returns a pointer on the public identifier (URN).

Syntax	**NSAX_GETPUBLICID (*P*)**			
Parameter	*P*	DYNSTR	I	Pointer on a "Locator"
Value returned	POINTER			

Comment

1. This pointer is a DynStr. If it is different to NULL, it should be then freed with NSAX_DISPOSE.

See also NSAX_GETCOLUMNNUMBER%, NSAX_GETLINENUMBER%, NSAX_GETSYSTEMID

NSAX_GETSYSTEMID function

Returns a pointer on the system identifier (URL).

Syntax	**NSAX_GETSYSTEMID (*P*)**			
Parameter	*P*	DYNSTR	I	Pointer on a "Locator"
Value returned	POINTER			

Comment

1. This pointer is a DynStr. If it is different to NULL, it should be then freed with NSAX_DISPOSE.

See also NSAX_GETCOLUMNNUMBER%, NSAX_GETLINENUMBER%, NSAX_GETPUBLICID

NSAX_PE_GETCOLUMNNUMBER% function

Returns the column number.

Syntax	**NSAX_PE_GETCOLUMNNUMBER% (*P*)**			
Parameter	*P*	POINTER	I/O	Pointer on an exception handle.
See also	NSAX_PE_GETLINENUMBER%, NSAX_PE_GETPUBLICID, NSAX_PE_GETSYSTEMID, NSAX_PE_GETMESSAGE, NSAX_DISPOSE			

NSAX_PE_GETLINENUMBER% function

Returns the row number.

Syntax	**NSAX_PE_GETLINENUMBER% (*P*)**			
Parameter	*P*	POINTER	I/O	Pointer on an exception handle.
See also	NSAX_PE_GETCOLUMNNUMBER%, NSAX_PE_GETPUBLICID, NSAX_PE_GETSYSTEMID, NSAX_PE_GETMESSAGE, NSAX_DISPOSE			

NSAX_PE_GETPUBLICID function

Returns a pointer on the public identifier (URN).

Syntax **NSAX_PE_GETPUBLICID (*P*)**

Parameter	*P*	DYNSTR	I/O	Pointer on an exception handle

Comment

1. This pointer is a DynStr. If it is different to NULL, it should be then freed with NSAX_DISPOSE.

See also NSAX_PE_GETCOLUMNNUMBER%, NSAX_PE_GETLINENUMBER%, NSAX_PE_GETSYSTEMID, NSAX_PE_GETMESSAGE, NSAX_DISPOSE

NSAX_PE_GETSYSTEMID function

Returns a pointer on the system identifier (URL).

Syntax **NSAX_PE_GETSYSTEMID (*P*)**

Parameter				
	P	DYNSTR	I/O	Pointer on an exception handle

Comment

1. This pointer is a DynStr. If it is different to NULL, it should be then freed with NSAX_DISPOSE.

See also NSAX_PE_GETCOLUMNNUMBER%, NSAX_PE_GETLINENUMBER%, NSAX_PE_GETPUBLICID, NSAX_PE_GETMESSAGE, NSAX_DISPOSE

NSAX_PE_GETMESSAGE function

Returns an error message

Syntax **NSAX_PE_GETMESSAGE (*P*)**

Parameter	*P*	DYNSTR	I/O	Pointer on an exception handle

Comment

1. This pointer is a DynStr. If it is different to NULL, it should be then freed with NSAX_DISPOSE.

See also NSAX_PE_GETCOLUMNNUMBER%, NSAX_PE_GETLINENUMBER%, NSAX_PE_GETPUBLICID, NSAX_PE_GETSYSTEMID, NSAX_DISPOSE

NSAX_DISPOSE instruction

This is used to free the memory allocated for *DynStr*s returned by the NSAX_GETPUBLICID, NSAX_GETSYSTEMID, NSAX_PE_GETPUBLICID, NSAX_PE_GETSYSTEMID and NSAX_PE_GETMESSAGE methods.

Syntax **NSAX_DISPOSE** *P*

Parameter

P	DYNSTR	I/O	Pointer on an exception handle

See also NSAX_PE_GETCOLUMNNUMBER%, NSAX_PE_GETLINENUMBER%, NSAX_PE_GETPUBLICID, NSAX_PE_GETSYSTEMID, NSAX_PE_GETMESSAGE

NSAX_INIFNT instruction

This informs the library of different methods to be called in callback.

Syntax **NSAX_INIFNT** *TY, P*

Parameters				
	TY	INTEGER	I	cSAXH_* constant that indicates the purpose of the method
	P	POINTER	I	pointer on the SAXH_* function to be implemented

Example

```
Instruction INITIALIZE_PARSER UNREPFLAGS%,EXPANDNAMESSPACES%(1)
Local Int RET

    NSAX_INIFNT cSAXH_WRITE,@SAXH_WRITE

    NSAX_INIFNT cSAXH_STARTDOCUMENT,@SAXH_STARTDOCUMENT
    NSAX_INIFNT cSAXH_ENDDOCUMENT,@SAXH_ENDDOCUMENT

    NSAX_INIFNT cSAXH_STARTELEMENT,@SAXH_STARTELEMENT
    NSAX_INIFNT cSAXH_ENDELEMENT,@SAXH_ENDELEMENT

    NSAX_INIFNT cSAXH_PROCESSINGINSTRUCTION,@SAXH_PROCESSINGINSTRUCTION
    NSAX_INIFNT cSAXH_COMMENT,@SAXH_COMMENT
    NSAX_INIFNT cSAXH_CHARACTERS,@SAXH_CHARACTERS

    NSAX_INIFNT cSAXH_ERROR,@SAXH_ERROR
    NSAX_INIFNT cSAXH_FATALERROR,@SAXH_FATALERROR
    NSAX_INIFNT cSAXH_WARNING,@SAXH_WARNING

    RET = NSAX_INITIALIZE%(UNREPFLAGS%,EXPANDNAMESSPACES%)

EndInstruction
```

See also cSAX_* constants, SAXH_STARTDOCUMENT, SAXH_ENDDOCUMENT, SAXH_STARTELEMENT, SAXH_ENDELEMENT, SAXH_PROCESSINGINSTRUCTION, SAXH_COMMENT, SAXH_CHARACTERS

cSAXH_* constants

This is used to indicate the purpose of the method called in callback by the NSAX_INIFNT instruction. .

Syntax **cSAXH_WRITE**

cSAXH_STARTDOCUMENT

cSAXH_ENDDOCUMENT

cSAXH_STARTELEMENT

cSAXH_ENDELEMENT

cSAXH_PROCESSINGINSTRUCTION

cSAXH_COMMENT

cSAXH_CHARACTERS

cSAXH_ERROR

cSAXH_FATALERROR

cSAXH_WARNING

Comment

1. They have the following meaning:

Text – *Reserved*

cSAXH_WRITE

Start/End of document

cSAXH_STARTDOCUMENT
cSAXH_ENDDOCUMENT

Elements

cSAXH_STARTELEMENT
cSAXH_ENDELEMENT

Instruction

cSAXH_PROCESSINGINSTRUCTION

Comment

cSAXH_COMMENT

Characters

cSAXH_CHARACTERS

Error management

cSAXH_ERROR
cSAXH_FATALERROR
cSAXH_WARNING

2. They are declared as follows:

```
cSAXH_WRITE                     1
cSAXH_STARTDOCUMENT             8
cSAXH_ENDDOCUMENT               9

cSAXH_STARTELEMENT             10
cSAXH_ENDELEMENT               11
cSAXH_PROCESSINGINSTRUCTION    12
cSAXH_COMMENT                  13
cSAXH_CHARACTERS               14

cSAXH_ERROR                    20
cSAXH_FATALERROR               21
cSAXH_WARNING                  22
```

Example

```
NSAX_INIFNT cSAXH_STARTDOCUMENT,@SAXH_STARTDOCUMENT
```

See also NSAX_INIFNT, SAXH_STARTDOCUMENT, SAXH_ENDDOCUMENT, SAXH_STARTELEMENT, SAXH_ENDELEMENT, SAXH_PROCESSINGINSTRUCTION, SAXH_COMMENT, SAXH_CHARACTERS

SAXH_STARTDOCUMENT function

This is called at the start of the syntax analysis.

This method is called by the parser (only once) when starting up the XML flow analysis. This is called before all other methods of the interface, except, obviously the setDocumentLocator method. This document should allow you to initialise everything that should be initialised, before starting to process the document

Syntax **SAXH_STARTDOCUMENT (***PDATA)*

Parameter	*PDATA*	POINTER	I	Information given via NSAX_CREATEPARSER

Value returned INT

Example

```
Global Control C_LST

FUNCTION SAXH_STARTDOCUMENT(POINTER PDATA) RETURN INT
    Insert At End " *** SAXH_STARTDOCUMENT *** " TO C_LST
    Return TRUE%
ENDFUNCTION
```

See also NSAX_CREATEPARSER, NSAX_INIFNT, SAXH_ENDDOCUMENT, SAXH_STARTELEMENT, SAXH_ENDELEMENT, SAXH_PROCESSINGINSTRUCTION, SAXH_COMMENT, SAXH_CHARACTERS

SAXH_ENDDOCUMENT function

This is called at the end of the syntax analysis.

This method is called at the end of the flow process after all other methods. It can then be useful when notifying other objects that the work has been finished.

Syntax **SAXH_ENDDOCUMENT (***PDATA)*

Parameter	*PDATA*	POINTER	I	Information given via NSAX_CREATEPARSER

Value returned INT

Example

```
Global Control C_LST

FUNCTION SAXH_ENDDOCUMENT(POINTER PDATA) RETURN INT
    Insert At End " *** SAXH_ENDDOCUMENT *** " TO C_LST
    Return TRUE%
ENDFUNCTION
```

See also NSAX_CREATEPARSER, NSAX_INIFNT, SAXH_STARTDOCUMENT, SAXH_STARTELEMENT, SAXH_ENDELEMENT, SAXH_PROCESSINGINSTRUCTION, SAXH_COMMENT, SAXH_CHARACTERS

SAXH_STARTELEMENT function

This is called at the beginning of an XML element analysis, when the start of an element is detected.

Syntax **SAXH_STARTELEMENT (***PDATA, NAME, ATTRIBUTES)*

Parameters

PDATA	POINTER	I	Information given via NSAX_CREATEPARSER
NAME	CSTRING	I	element name
ATTRIBUTES	POINTER	I/O	pointer on the list of the elements' attributes

Comment

1. The *ATTRIBUTES* parameter is used with the NSAX_ATTRIBUTE_* functions to list the set of attributes and their values.

Example

```
Global Control C_LST
Global Control C_LERR
Global POINTER P_LOC
Global Line$(2048)


Segment S_PRIVDATA
POINTER LOCATOR
EndSegment

Segment S_POINTER
Pointer P
EndSegment
Segment S_CSTRING
CSTRING S
EndSegment

Segment S_DYNSTR
DYNSTR DS
EndSegment

FUNCTION SAXH_STARTELEMENT(POINTER PDATA,CSTRING NAME,POINTER \
@ATTRIBUTES) RETURN INT
Local LINE$,I%,N%,Pointer P,ILINE%

ILINE% = NSAX_GETLINENUMBER%(S_PRIVDATA(PDATA).LOCATOR)
LINE$ = "Line=" & ILINE% && "=<" & NAME
Insert At End LINE$ TO C_LST

N% = NSAX_ATTRIBUTES_GETLENGTH%(ATTRIBUTES)-1
    If N% >= 0
    For I%=0 TO N%
            LINE$ = "    id:" & I%
```

```
            P = NSAX_ATTRIBUTE_GETNAME%(ATTRIBUTES,I%,TRUE%)
            If P
                   LINE$ = LINE$ && "NAME=" & S_CSTRING(P).S
                   NSAX_DISPOSE P
            Else
                   LINE$ = LINE$ && "NAME=''"
            EndIf
            P = NSAX_ATTRIBUTE_GETTYPE%(ATTRIBUTES,I%)
            If P
                   LINE$ = LINE$ && "TYPE=" & S_CSTRING(P).S
                   NSAX_DISPOSE P
            Else
                   LINE$ = LINE$ & "TYPE=''"
            EndIf
            P = NSAX_ATTRIBUTE_GETVALUE%(ATTRIBUTES,I%)
            If P
                   LINE$ = LINE$ && "VALUE=" & S_CSTRING(P).S
                   NSAX_DISPOSE P
            Else
                   LINE$ = LINE$ & "VALUE=''"
            EndIf
            Insert At End LINE$ TO C_LST
    EndFor
    Insert At End "..." & NAME && ">" TO C_LST
    EndIf

Return TRUE%

ENDFUNCTION
```

See also NSAX_CREATEPARSER, NSAX_INIFNT, SAXH_STARTDOCUMENT, SAXH_ENDDOCUMENT, SAXH_ENDELEMENT, SAXH_PROCESSINGINSTRUCTION, SAXH_COMMENT, SAXH_CHARACTERS, NSAX_ATTRIBUTE_*

SAXH_ENDELEMENT function

This is called at the end of an XML element analysis, when the end of an element is detected.

Syntax **SAXH_ENDELEMENT (***PDATA, NAME)*

Parameter	*PDATA*	POINTER	I	Information given via NSAX_CREATEPARSER
	NAME	CSTRING	I	element name

Value returned INT

Example

```
Global Control C_LST

FUNCTION SAXH_ENDELEMENT(POINTER PDATA,CSTRING NAME) RETURN INT
    Insert At End "</"&NAME&">" TO C_LST
Return TRUE%
ENDFUNCTION
```

See also NSAX_CREATEPARSER, NSAX_INIFNT, SAXH_STARTDOCUMENT, SAXH_ENDDOCUMENT, SAXH_STARTELEMENT, SAXH_PROCESSINGINSTRUCTION, SAXH_COMMENT, SAXH_CHARACTERS

SAXH_PROCESSINGINSTRUCTION function

Processing instruction node in the output.

This event is called for each functioning instruction found. These instructions are those that you find outside of the XML tree itself.

Syntax **SAXH_PROCESSINGINSTRUCTION (***PDATA, TARGET, DATA)*

Parameters

PDATA	POINTER	I	Information given via NSAX_CREATEPARSER
TARGET	DYNSTR	I/O	target
DATA	DYNSTR	I/O	data

Example

```
Global Control C_LST

FUNCTION SAXH_PROCESSINGINSTRUCTION(POINTER PDATA,DynStr @TARGET,DynStr
@DATA) RETURN INT
    Insert At End "*** PROCESSINGINSTRUCTION Target:" && TARGET &&  \
      "Data:" && DATA TO C_LST
    Return TRUE%
ENDFUNCTION
```

See also NSAX_CREATEPARSER, NSAX_INIFNT, SAXH_STARTDOCUMENT, SAXH_ENDDOCUMENT, SAXH_STARTELEMENT, SAXH_ENDELEMENT, SAXH_COMMENT, SAXH_CHARACTERS

SAXH_COMMENT function

This is called when a comment is found, i.e. text between "<!—" and "-->". For example; <!-- Edited with XML Spy v2007 (http://www.altova.com) -->.

Syntax **SAXH_COMMENT (***PDATA, ds)*

Parameters

PDATA	POINTER	I	information given via NSAX_CREATEPARSER
ds	DYNSTR	I/O	comment

Example

```
Global Control C_LST
Global Control C_LERR
Global POINTER P_LOC
Global Line$(2048)

Segment S_PRIVDATA
POINTER LOCATOR
EndSegment

Segment S_CSTRING
CSTRING S
EndSegment


FUNCTION SAXH_COMMENT(POINTER PDATA,DynStr @ds) RETURN INT
Local P%
      Insert At End "Line=" &
NSAX_GETLINENUMBER%(S_PRIVDATA(PDATA).LOCATOR) && " Comment:" && ds TO
C_LST
      P% = NSAX_GETSYSTEMID(S_PRIVDATA(PDATA).LOCATOR)
      If P%
      Insert At End "NSAX_GETSYSTEMID" && S_CSTRING(P%).S TO C_LST
             NSAX_DISPOSE P%
      EndIf
      P% = NSAX_GETPUBLICID(S_PRIVDATA(PDATA).LOCATOR)
      If P%
      Insert At End "NSAX_GETPUBLICID" && S_CSTRING(P%).S TO C_LST
             NSAX_DISPOSE P%
      EndIf
      Return TRUE%
ENDFUNCTION
```

See also NSAX_CREATEPARSER, NSAX_INIFNT, SAXH_STARTDOCUMENT, SAXH_ENDDOCUMENT, SAXH_STARTELEMENT, SAXH_ENDELEMENT, SAXH_PROCESSINGINSTRUCTION, SAXH_CHARACTERS

SAXH_CHARACTERS function

Detecting data between two elements triggers a call from this event.

Syntax **SAXH_CHARACTERS (***PDATA, src)*

Parameters				
	PDATA	POINTER	I	information given via NSAX_CREATEPARSER
	src	DYNSTR	I/O	source

Value returned INT

Comment

1. In general, this event is called simply by the presence of text between opening and closing elements, such as in this example:

```
<myTag>some text</myTag>
```

In the example below (chars.xml), SAXH_CHARACTERS will be called three times, for “some” then “scattered” and finally “text”

```
<myTag>some
    <embeddedTag name="hi"/> scattered<embeddedTag name="a
no"/>text
</myTag>
```

Example

```
Global Control C_LST

FUNCTION SAXH_CHARACTERS(POINTER PDATA,DynStr @src) RETURN INT
Local P%, DynStr dst, Char C, I%

    If Length(src) > 0

        dst = ''
        For I%=1 To Length(src)
            C = copy$(src,I%,1)
            If ASC%(C) = 10
                dst = dst & '<RC>'
            ElseIf ASC%(C) = 9
                dst = dst & '<TAB>'
            ElseIf ASC%(C) < 32
                dst = dst & '<' & ASC%(C) &'>'
            Else
                dst = dst & C
            EndIf
        EndFor

        Insert At End "Line=" &
NSAX_GETLINENUMBER%(S_PRIVDATA(PDATA).LOCATOR) && 'Length:' &&
Length(src) && " Characters:" && dst TO C_LST
    EndIf
```

```
    Return TRUE%

ENDFUNCTION
```

See also NSAX_CREATEPARSER, NSAX_INIFNT, SAXH_STARTDOCUMENT, SAXH_ENDDOCUMENT, SAXH_STARTELEMENT, SAXH_ENDELEMENT, SAXH_PROCESSINGINSTRUCTION, SAXH_COMMENT

SAXH_ERROR% function

Returns an error code.

Syntax **SAXH_ERROR% (***PDATA, E)*

Parameters				
	PDATA	POINTER	I	information given via NSAX_CREATEPARSER
	E	POINTER	I/O	Handle to be used with the NSAX_PE_* methods

Comment

1. If TRUE% is returned, the system resets the error code to zero.

Example

```
Global Control C_LERR

Instruction ErrorMsg POINTER @E
Local Pointer P
    P = NSAX_PE_GETSYSTEMID(E)
    If P
            Insert At End "NSAX_PE_GETSYSTEMID:"&&S_CSTRING(P).S TO
C_LERR
            NSAX_DISPOSE P
    EndIf
    P = NSAX_PE_GETPUBLICID(E)
    If P
            Insert At End "NSAX_PE_GETPUBLICID:"&&S_CSTRING(P).S TO
C_LERR
            NSAX_DISPOSE P
    EndIf
    P = NSAX_PE_GETMESSAGE(E)
    If P
            Insert At End "NSAX_PE_GETMESSAGE:"&&S_CSTRING(P).S TO
C_LERR
            NSAX_DISPOSE P
    EndIf
EndInstruction

FUNCTION SAXH_ERROR(POINTER PDATA, POINTER @E) RETURN INT
    Insert At End "SAXH_ERROR"&&"Col:"&NSAX_PE_GETCOLUMNNUMBER%(E)&"
Line:"&NSAX_PE_GETLINENUMBER%(E) TO C_LERR
    ErrorMsg(E)
    Return TRUE%
```

See also NSAX_CREATEPARSER, SAXH_FATALERROR%, SAXH_WARNING%, NSAX_PE_*

SAXH_FATALERROR% function

Returns a fatal error.

Syntax **SAXH_FATALERROR% (***PDATA, E)*

Parameters				
	PDATA	POINTER	I	information given via NSAX_CREATEPARSER
	E	POINTER	I/O	Handle to be used with the NSAX_PE_* methods

Comment

1. If TRUE% is returned, the system resets the error code to zero.

Example

```
Global Control C_LERR

Instruction ErrorMsg POINTER @E
Local Pointer P
P = NSAX_PE_GETSYSTEMID(E)
  If P
          Insert At End "NSAX_PE_GETSYSTEMID:"&&S_CSTRING(P).S TO C_LERR
NSAX_DISPOSE P
 EndIf
P = NSAX_PE_GETPUBLICID(E)
  If P
          Insert At End "NSAX_PE_GETPUBLICID:"&&S_CSTRING(P).S TO C_LERR
NSAX_DISPOSE P
 EndIf
P = NSAX_PE_GETMESSAGE(E)
  If P
           Insert At End "NSAX_PE_GETMESSAGE:"&&S_CSTRING(P).S TO C_LERR
NSAX_DISPOSE P
 EndIf
EndInstruction

FUNCTION SAXH_FATALERROR(POINTER PDATA, POINTER @E) RETURN INT
      Insert At End
"SAXH_FATALERROR"&&"Col:"&NSAX_PE_GETCOLUMNNUMBER%(E)&"
Line:"&NSAX_PE_GETLINENUMBER%(E) TO C_LERR
      ErrorMsg(E)
      Return TRUE%
ENDFUNCTION
```

See also NSAX_CREATEPARSER, SAXH_ERROR%, SAXH_WARNING%, NSAX_PE_*

SAXH_WARNING% function

Returns a warning.

Syntax **SAXH_WARNING% (***PDATA, E)*

Parameters				
	PDATA	POINTER	I	information given via NSAX_CREATEPARSER
	E	POINTER	I/O	Handle to be used with the NSAX_PE_* methods

Comment

1. If TRUE% is returned, the system resets the error code to zero.

Example

```
Global Control C_LERR

Instruction ErrorMsg POINTER @E
Local Pointer P
P = NSAX_PE_GETSYSTEMID(E)
  If P
          Insert At End "NSAX_PE_GETSYSTEMID:"&&S_CSTRING(P).S TO C_LERR
NSAX_DISPOSE P
 EndIf
P = NSAX_PE_GETPUBLICID(E)
  If P
          Insert At End "NSAX_PE_GETPUBLICID:"&&S_CSTRING(P).S TO C_LERR
NSAX_DISPOSE P
 EndIf
P = NSAX_PE_GETMESSAGE(E)
  If P
           Insert At End "NSAX_PE_GETMESSAGE:"&&S_CSTRING(P).S TO C_LERR
NSAX_DISPOSE P
 EndIf
EndInstruction

FUNCTION SAXH_WARNING(POINTER PDATA, POINTER @E) RETURN INT
Message "SAXH_WARNING","Col:"&NSAX_PE_GETCOLUMNNUMBER%(E)&"
Line:"&NSAX_PE_GETLINENUMBER%(E)
ErrorMsg(E)
Return TRUE%
ENDFUNCTION
```

See also NSAX_CREATEPARSER, SAXH_ERROR%, SAXH_FATALERROR%, NSAX_PE_*