F77xml 1.0 library documentation

• Introduction

The f77xml/f90xml library is designed to interface libgdome2, which can be found at http://gdome2.cs.unibo.it/. Most of the design is implemented to give access to a full DOM level 2 interface (with the exclusion of events, which are hopefully not needed in the target environment).

• Design

The library has been designed with f77 backward compatibility in mind. Although fortran 77 is very old and the choice for new code should be fortran 90 (or other languages), this choice was made to provide full freedom even to the projects that still are stuck with pure f77 for their maintenance (because of g77, although the situation should change with gcc 4.0, which includes a f95 compiler), but want to implement an xml interface. Keep in mind that, although feasible, the fortran 77 access to the library is everything but simple to use. A wrapper could simplify the work, but this is not my priority at the moment.

The main concepts are:

Standard call structure the call structure is standardized as follows: all gdome2 functions are mapped to f90 subroutines (so you need to call them with CALL). The first argument is the return value from the gdome2 routine, so it is an INTENT(OUT) parameter. The subsequent arguments are the requested gdome2 parameters, so they are INTENT(IN), except the error which is an INTENT(OUT) and is always at the end of the parameter list. If the gdome2 function returns void, the f77xml/f90xml routine accepts an integer as a first parameter, but the returned value is meaningless.

multiplexers: In a perfect world, f77 supports long names for routines and namespacing. In our real world, this is not true. The standard f77 expect to have names no longer than 6 characters. Even if no recent f77 compiler imposes this, I like to be compliant (in the limits of my knowledge, I have not so much time and I prefer to focus on results rather than philosophy) so the f77 interface is designed to provide subroutines of 6 characters or less. Given that the DOM compliant interface holds hundreds of functions, the choice of implementing an arbitrary combination of characters to create every function would result in a incomprehensible mess of letters with poor or no meaning at all. Also, the space of available combinations is rather limited, and this could result in name collision between different subroutines of different libraries which happen to have the same name and be linked to the same executable. For solving these problems, a few multipurpose functions have been created, named multiplexers. They have a fixed name, depending on number and type of parameters, and the long name for the final function to call is passed as a character parameter. Internally, the f77xml library uses this character parameter to choose which gdome function needs to be called. For example, the gdome function

```
GdomeNode*  gdome_el_firstChild  (GdomeElement *self, GdomeException *exc);
```

is classified as a function that accepts 2 parameters (the GdomeElement and the GdomeException) returning a GdomeNode.
The multiplexer associated is therefore xp3t1, “x” is a standard prefix, p3 means 3 parameters (2 parameters + returned object. More on this later) and t1 means parameter type 1. The type distinguish routines that accept 3 parameters but of different kinds. For example

```
GdomeBoolean  gdome_el_hasChildNodes  (GdomeElement *self, GdomeException *exc);
```

this function accepts a GdomeElement and a GdomeException and returns a boolean value, so is handled by xp3t4, 3 parameters type 4). The choice of the type number cannot in general be derived.
So how to call functions from f77? Here is the code

```
character*128 func
integer :: node,elem,err
func='el_firstChild'
CALL xp3t1(node,func,elem,err)
```

As you can see, it' s qite tricky, because you need to know how to map every function to the corresponding multiplexer. This is provided in the appendix of this document. The func name is case sensitive and is the exact copy of the gdome2 function, stripped of the “gdome_” prefix. From f90 this is not an issue, since you do not need to specify the multiplexer by hand. F90 support long names for routines, and the mapping is already implemented in the library. You then need to call

```
integer :: node,elem,err
CALL f90xml_el_firstChild(node,elem,err)
```

Objects mapped to integers every object (a Node, a DOMImplementation, a DOMString etc...) is mapped to an integer, and from the fortran side you handle these integers. In the chosen formalism, these integers are named **codes**. From the practical point of view, these are indexes into a hash table to identify an object and recover its pointer, skipping the tedious handling of 32 bit memory pointers on the fortran size.
In my idea, you don't need to clean these objects, since a primitive garbage collector (note of warning: technically it isn')tcan be manually triggered to clean up all the allocated structures. There's no way to know if some integer is still used in the fortran code (indeed they can be taken out of thin air, if you want, but expect weird results) so I'm not able to figure out how to clean only unused (ie: no more referenced) nodes keeping the used ones. Take this technique as a strategy to delay the cleanup of many objects to a final single call. However, **functions for unref the structures (Nodes, DOMStrings etc...) are available, and you are encouraged to use them**. The automatic cleanup will be implemented if i have time.

Code example:

```
f90:
integer :: elem,doc,err
CALL f90xml_doc_documentElement(elem,doc,err)

f77:
character*128 func
integer :: elem,doc,err
func='doc_documentElement'
CALL xp3t1(elem,func,doc,err)
```

in this call, elem and doc are all integers. err is an integer, but it references an error code, not an object, of course.

Signatures: functions that are referenced by the same multiplexer accept the same number and type of parameters. A convenient way to depict the characteristic of the parameter list is with a signature in the form (return|parameters). An example will be clearer. The subroutine f90xml_doc_documentElement accepts a document and return an element. The signature of this function is (c|ce), because the underlying C routine returns a code, and accepts a code and an error value to write via a pass-by-reference. The most convenient way to understand signatures is to compare the gdome2 routines and the f90xml routines.
For each kind of parameter a letter has been assigned “c” for code, “b” for boolean (logical), “e” for error, “s” for string, “u” for unsigned integer. The resulting signature is used to get the name of the multiplexer, so a (c|ce) is handled by a xp3 kind of multiplexer. For the type, at the moment there' sno exact match, although this could change in the future, but for the moment you need to know that the type for (c|ce) is 1 and so the multiplexer that handles doc_documentElement is xp3t1.

Strings: where signatures hold an “s” letter, a character(len=*) value is required. For most of the time, strings in f77/f90xml are handled as DOMString objects and then if you query for an element name, for example, you obtain back a code that references to a string object. Also, when you need to set a name, or data of a textNode, you need to pass a DOMString object, thus a code, and not a character fortran parameter.
To move back and forth between the DOMString object concept and the character concept you have some useful subroutines:

f90xml_str_mkref (signature : p3t2 (c|se))
creates a new string using a character fortran string. Returns a code that references to the DOMString object.

f90xml_str_length (signature : p3t3 (u|ce))
accepts the code of the DOMString and returns the length of this string. It's useful in case you want to know in advance how many bytes you need to read, so if it returns 300 and you have a character*100 you know how to deal with it.

f90xml_str_toFortran (signature p5t2 (b|csue))
This function is used to extract fortran character data from an existing DOMString object. The parameters accepted are the code of the DOMString, the character(len=*) variable which, at the function return will hold the characters, an unsigned int value which specifies the starting offset for extracting data from the DOMString and the error. The boolean parameter returned is true if the extraction procedure runs up to the end of the string. More on this in the documentation of the subroutine.

f90xml_str_unref (signature p3t1 (v|ce))
delete the DOMString, and if there are no other references to this object, remove code from the cache.

- Errors

The f77/f90xml library returns errors as integers. They are reported here, along with a comment.

0 : ERR_NO_ERROR
no error. Everything ok.

10 : ERR_DATA_NOT_AN_ELEMENT
11 : ERR_DATA_NOT_A_NODE
12 : ERR_DATA_NOT_A_DOCUMENT
13 : ERR_DATA_NOT_A_STRING
14 : ERR_DATA_NOT_A_NODELIST
15 : ERR_DATA_NOT_A_COMMENT
16 : ERR_DATA_NOT_AN_ATTR
17 : ERR_DATA_NOT_A_NAMEDNODEMAP
18 : ERR_DATA_NOT_A_TEXT
19 : ERR_DATA_NOT_AN_ENTITYREF
20 : ERR_DATA_NOT_AN_ENTITY
21 : ERR_DATA_NOT_A_PROCESSINGINSTRUCTION
22 : ERR_DATA_NOT_A_CDATASECTION
23 : ERR_DATA_NOT_A_DOCUMENTFRAGMENT
24 : ERR_DATA_NOT_A_DOCUMENTTYPE
25 : ERR_DATA_NOT_A_DOMIMPLEMENTATION
These errors are returned every time a passed code is not of the expected type, after inquiry into the cache. In other words, if you pass the code that corresponds to a DOMImplementation to f90xml_el_firstChild, you are returned with *ERR_DATA_NOT_AN_ELEMENT*, because f90xml_el_firstChild expects an element. Be warned that if you have a code that refers to an element you **can** pass it to f90xml_n_whatever (which accept a code for a node), because an element is also a node.

30 : ERR_NO_CACHE_HIT
You obtain this error every time you pass a code that does not reference to any object in the cache, for example when the code is picked out from thin air, or the internal object that was referred with that code has been deleted from the cache.

31 : ERR_NULL_CODE
This error is returned every time you pass a null code (code equal to zero) to a routine.

1000 : ERR_NEVER_RETURN_THIS
if you obtain this error, something very bad has happened. Try to reproduce the problem with a small program and contact the author.

10000: ERR_GDOME
An error at gdome2 level has been detected. There are various reasons for this to happen. See above.

Types
DOM defines a standard numeration for type of nodes. This value can be accessed with the nodeType kind of functions. The GDOME library match the standard enumeration

```
GDOME_ELEMENT_NODE = 1
GDOME_ATTRIBUTE_NODE = 2
GDOME_TEXT_NODE = 3
GDOME_CDATA_SECTION_NODE = 4
GDOME_ENTITY_REFERENCE_NODE = 5
GDOME_ENTITY_NODE = 6
GDOME_PROCESSING_INSTRUCTION_NODE = 7
GDOME_COMMENT_NODE = 8
GDOME_DOCUMENT_NODE = 9
GDOME_DOCUMENT_TYPE_NODE = 10
GDOME_DOCUMENT_FRAGMENT_NODE = 11
GDOME_NOTATION_NODE = 12
GDOME_XPATH_NAMESPACE_NODE = 13
```

In the same way, the F90 module defines these variables

```
integer,parameter :: F90XML_ELEMENT_NODE = 1
integer,parameter :: F90XML_ATTRIBUTE_NODE = 2
integer,parameter :: F90XML_TEXT_NODE = 3
integer,parameter :: F90XML_CDATA_SECTION_NODE = 4
integer,parameter :: F90XML_ENTITY_REFERENCE_NODE = 5
integer,parameter :: F90XML_ENTITY_NODE = 6
integer,parameter :: F90XML_PROCESSING_INSTRUCTION_NODE = 7
integer,parameter :: F90XML_COMMENT_NODE = 8
integer,parameter :: F90XML_DOCUMENT_NODE = 9
integer,parameter :: F90XML_DOCUMENT_TYPE_NODE = 10
integer,parameter :: F90XML_DOCUMENT_FRAGMENT_NODE = 11
integer,parameter :: F90XML_NOTATION_NODE = 12
integer,parameter :: F90XML_XPATH_NAMESPACE_NODE = 13
```

Null Code

The following variables are defined:

```
integer,parameter :: F90XML_NULLCODE = 0
integer,parameter :: NullCode = 0
```

The first is more namespaced, while the second is more readable in code. I think

```
if (code == NullCode) then
```

is more readable than

```
if (code == F90XML_NULLCODE) then
```

use at your discretion.

- F77 interface

The fortran 77 interface is built on variuos multiplexers. They are listed here

```
LOGICAL :: boolValue
INTEGER :: intValue1,intValue2
INTEGER  :: code,code1,code2,code3,code4,code5
CHARACTER(len=*) :: funcName
CHARACTER(len=*) :: string
INTEGER :: error
```

```
void xp2t1(code, funcName, error)
void xp3t1(code, funcName, code1, error)
void xp3t2(code, funcName, string, error)
void xp3t3(intValue, funcName, code1, error)
void xp3t4(boolValue, funcName, code1, error)
void xp4t1(code, funcName, code1, code2, error)
void xp4t2(code, funcName, code1, intValue, error)
void xp4t3(boolValue, funcName, code1, code2, error)
void xp4t4(code, funcName, code1, boolValue, error)
void xp5t1(code, funcName, code1, code2, code3, error)
void xp5t2(boolValue, funcName, code1, string, intValue1, error)
void xp5t3(boolValue, funcName, code1, code2, code3, error)
void xp5t4(code, funcName, code1, intValue1, intValue2, error)
void xp5t5(code, funcName, code1, string, intValue1, error)
void xp5t6(code, funcName, code1, intValue, code2, error)
void xp5t7(code, funcName, code1, code2, boolValue, error)
void xp6t1(code, funcName, code1, code2, code3, code4, error)
void xp6t2(boolValue, funcName, code1, code2, string, intValue, error)
void xp6t3(code, funcName, code1, intValue1, intValue2, code2,  error)
```

- F90 interface

All the interface is defined into the f90xml module. You need to include the f90 statement

```
use f90xml
```

to enable the search and inclusion of the .mod file.
Due to the high similarity between f90xml and gdome2 routine, documenting the whole library is highly redundant. For a reference see http://gdome2.cs.unibo.it/gtk-doc/book1.html. Instead, a systematic analysis of differences between f90xml and gdome2 will be performed.

DomImplementation:

functions not implemented

gdome_di_ref
gdome_di_query_interface
gdome_di_hasFeature
gdome_di_freeDoc
gdome_di_createDocFromMemory
gdome_di_createDocFromURIWithEntitiesTable
gdome_di_createDocFromMemoryWithEntitiesTable
gdome_di_saveDocToFileEnc
gdome_di_saveDocToMemory
gdome_di_saveDocToMemoryEnc

functions with a different implementation:

- GdomeDocument * gdome_di_createDocFromURI (GdomeDOMImplementation *self, const char *uri, unsigned int mode, GdomeException *exc);

the parameter mode must be chosen accordingly on this table

0 : GDOME_LOAD_PARSING
1 : GDOME_LOAD_VALIDATING
2 : GDOME_LOAD_RECOVERING

additional flags that can be passed together:
4 : GDOME_LOAD_SUBSTITUTE_ENTITIES
8 : GDOME_LOAD_COMPLETE_ATTR

which must be summed to the chosen mode (technically is an OR operation). So if you want to do validation and substitute entities, then pass 5 as mode.
Using this function with F90xml imposes these caveats:

- In the F90 module, the defines given above are replaced by variables defined in the module and can be used. Thus in F90 you can use F90XML_LOAD_PARSING, for example, and its value is 0.

- The filename URI must be passed with a variable, not inline.

  ```
  character(len=128) :: uri = “file.xml”
  call f90xml_di_createDocFromURI(doc,domimpl,uri, F90XML_LOAD_PARSING, err)
  ```

  is correct, while

  ```
  call f90xml_di_createDocFromURI(doc,domimpl,”file.xml”, F90XML_LOAD_PARSING, err)
  ```

  is not.

- GdomeBoolean gdome_di_saveDocToFile (GdomeDOMImplementation *self, GdomeDocument *doc, const char *filename, GdomeSavingCode mode, GdomeException *exc);

GdomeSavingCode can be

0 : GDOME_SAVE_STANDARD
1 : GDOME_SAVE_LIBXML_INDENT

The F90 module defines the same identifiers as variables, defined with the prefix F90XML as discussed above. In other words, these variables are defined

0: F90XML_SAVE_STANDARD
1: F90XML_SAVE_LIBXML_INDENT

- Final remarks

- Appendix

signature list

signature p2t1 (c|e)

di_mkref

signature p3t1 (c|ce)

doc_documentElement	doc_implementation	el_firstChild
el_lastChild	el_nextSibling	el_previousSibling
el_parentNode	er_parentNode	dt_parentNode
df_parentNode	pi_firstChild	pi_lastChild
pi_nextSibling	pi_previousSibling	pi_parentNode
cd_firstChild	cd_lastChild	cd_nextSibling
cd_previousSibling	cd_parentNode	cds_parentNode
not_firstChild	not_lastChild	not_nextSibling
not_previousSibling	not_parentNode	t_firstChild
t_lastChild	t_nextSibling	t_previousSibling
t_parentNode	doc_firstChild	doc_lastChild
doc_nextSibling	doc_previousSibling	doc_parentNode
el_childNodes	er_childNodes	dt_childNodes
df_childNodes	pi_childNodes	cd_childNodes
not_childNodes	cds_childNodes	t_childNodes
doc_childNodes	a_childNodes	el_tagName
el_nodeName	n_nodeName	ent_nodeName
c_nodeName	doc_nodeName	er_nodeName
dt_nodeName	not_nodeName	df_nodeName
pi_nodeName	cd_nodeName	cds_nodeName
a_nodeName	t_nodeName	el_nodeValue
df_nodeValue	pi_nodeValue	cd_nodeValue
not_nodeValue	t_nodeValue	doc_nodeValue
cds_nodeValue	el_prefix	el_namespaceURI
el_ownerDocument	er_ownerDocument	dt_ownerDocument
df_ownerDocument	pi_ownerDocument	cd_ownerDocument
not_ownerDocument	n_ownerDocument	c_ownerDocument
ent_ownerDocument	doc_ownerDocument	t_ownerDocument
a_ownerDocument	cds_ownerDocument	el_localName
er_localName	dt_localName	df_localName
pi_localName	cd_localName	not_localName
cds_localName	a_localName	doc_localName
el_attributes	pi_attributes	cd_attributes
not_attributes	a_attributes	t_attributes
doc_attributes	n_nodeValue	n_parentNode
n_childNodes	n_firstChild	n_lastChild
n_previousSibling	n_nextSibling	er_firstChild
er_lastChild	er_previousSibling	er_nextSibling
df_firstChild	df_lastChild	df_previousSibling
df_nextSibling	dt_firstChild	dt_lastChild
dt_previousSibling	dt_nextSibling	cds_firstChild
cds_lastChild	cds_previousSibling	cds_nextSibling
n_namespaceURI	n_prefix	n_localName
c_data	pi_data	cd_data
cds_data	t_data	c_nodeValue
c_parentNode	c_childNodes	c_firstChild
c_lastChild	c_previousSibling	c_nextSibling

c_namespaceURI	c_prefix	c_localName
n_attributes	er_attributes	dt_attributes
df_attributes	cds_attributes	doc_createDocumentFragment
el_normalize	er_normalize	dt_normalize
df_normalize	pi_normalize	cd_normalize
not_normalize	cds_normalize	a_normalize
t_normalize	doc_normalize	n_normalize
c_normalize	ent_normalize	c_attributes
doc_doctype	ent_publicId	ent_systemId
dt_publicId	dt_systemId	dt_internalSubset
not_publicId	not_systemId	ent_notationName
ent_nodeValue	er_nodeValue	dt_nodeValue
ent_firstChild	ent_lastChild	ent_nextSibling
ent_previousSibling	ent_parentNode	ent_childNodes
ent_attributes	ent_namespaceURI	er_namespaceURI
dt_namespaceURI	df_namespaceURI	pi_namespaceURI
cd_namespaceURI	not_namespaceURI	cds_namespaceURI
a_namespaceURI	t_namespaceURI	doc_namespaceURI
ent_prefix	er_prefix	dt_prefix
df_prefix	pi_prefix	cd_prefix
not_prefix	cds_prefix	a_prefix
t_prefix	doc_prefix	ent_localName
t_localName	a_name	a_ownerElement
a_value	a_nodeValue	a_firstChild
a_lastChild	a_nextSibling	a_previousSibling
a_parentNode	pi_target	dt_name
dt_entities	dt_notations	n_unref
el_unref	t_unref	cd_unref
cds_unref	c_unref	di_unref
doc_unref	t_unref	ent_unref
er_unref	nnm_unref	nl_unref
not_unref	pi_unref	a_unref
str_unref		

signature p3t2 (c|se)

str_new

signature p3t3 (u|ce)

str_len	str_length	nl_length
nnm_length	n_nodeType	er_nodeType
dt_nodeType	df_nodeType	pi_nodeType
cd_nodeType	not_nodeType	cds_nodeType
a_nodeType	t_nodeType	el_nodeType
c_nodeType	ent_nodeType	doc_nodeType
c_length	t_length	cds_length
cd_length		

signature p3t4 (b|ce)

el_hasChildNodes	er_hasChildNodes	dt_hasChildNodes
df_hasChildNodes	pi_hasChildNodes	cd_hasChildNodes
not_hasChildNodes	cds_hasChildNodes	a_specified
a_hasChildNodes	t_hasChildNodes	doc_hasChildNodes
el_hasAttributes	er_hasAttributes	dt_hasAttributes
df_hasAttributes	pi_hasAttributes	cd_hasAttributes
not_hasAttributes	cds_hasAttributes	a_hasAttributes

t_hasAttributes	doc_hasAttributes	n_hasChildNodes
n_hasAttributes	c_hasAttributes	c_hasChildNodes
ent_hasChildNodes	ent_hasAttributes	

signature p4t1 (c|cce)

nnm_getNamedItem	nnm_setNamedItem	nnm_setNamedItemNS
nnm_removeNamedItem	el_appendChild	a_set_value
doc_createElement	doc_createTextNode	doc_createComment
el_getAttribute	el_removeAttribute	el_getElementsByTagName
el_getAttributeNode	el_setAttributeNode	el_removeAttributeNode
el_setAttributeNodeNS	el_removeChild	er_removeChild
dt_removeChild	df_removeChild	pi_removeChild
cd_removeChild	not_removeChild	cds_removeChild
doc_removeChild	a_removeChild	er_set_prefix
el_set_prefix	dt_set_prefix	df_set_prefix
pi_set_prefix	cd_set_prefix	not_set_prefix
cds_set_prefix	a_set_prefix	el_set_nodeValue
r_set_nodeValue	dt_set_nodeValue	df_set_nodeValue
cd_set_nodeValue	a_set_nodeValue	doc_set_nodeValue
n_appendChild	er_appendChild	dt_appendChild
df_appendChild	pi_appendChild	cd_appendChild
not_appendChild	cds_appendChild	a_appendChild
doc_appendChild	t_appendChild	n_removeChild
t_removeChild	n_set_prefix	doc_set_prefix
n_set_nodeValue	not_set_nodeValue	c_set_data
pi_set_data	cd_set_data	cds_set_data
t_set_data	c_appendData	cd_appendData
cds_appendData	t_appendData	c_removeChild
c_appendChild	c_set_nodeValue	pi_set_nodeValue
cds_set_nodeValue	c_set_prefix	t_set_nodeValue
t_set_prefix	doc_createCDATASection	doc_createAttribute
doc_createEntityReference	doc_getElementsByTagName	ent_set_nodeValue
ent_removeChild	ent_appendChild	ent_set_prefix
doc_getElementById		

signature p4t2 (c|cue)

nl_item	nnm_item	t_splitText
cds_splitText		

signature p4t3 (b|cce)

el_canAppend	er_canAppend	dt_canAppend
df_canAppend	pi_canAppend	cd_canAppend
not_canAppend	cds_canAppend	a_canAppend
t_canAppend	doc_canAppend	n_canAppend
c_canAppend	ent_canAppend	

signature p4t4 (c|cbe)

el_cloneNode	n_cloneNode	c_cloneNode
doc_cloneNode	t_cloneNode	a_cloneNode
cds_cloneNode	cd_cloneNode	pi_cloneNode
df_cloneNode	not_cloneNode	dt_cloneNode
ent_cloneNode	er_cloneNode	

signature p5t1 (c|ccce)

nnm_getNamedItemNS	nnm_removeNamedItemNS	el_setAttribute
doc_createElementNS	doc_createAttributeNS	doc_getElementsByTagNameNS
el_getAttributeNS	el_getAttributeNodeNS	el_getElementsByTagNameNS
el_insertBefore	er_insertBefore	dt_insertBefore
df_insertBefore	pi_insertBefore	cd_insertBefore
not_insertBefore	el_replaceChild	er_replaceChild
dt_replaceChild	df_replaceChild	pi_replaceChild
cd_replaceChild	not_replaceChild	cds_insertBefore
cds_replaceChild	t_insertBefore	t_replaceChild
doc_insertBefore	doc_replaceChild	n_insertBefore
n_replaceChild	c_insertBefore	c_replaceChild
ent_insertBefore	ent_replaceChild	a_insertBefore
a_replaceChild	el_removeAttributeNS	
doc_createProcessingInstruction		

signature p5t2 (b|csue)

str_toFortran

signature p5t3 (b|ccce)

el_hasAttributeNS

signature p5t4 (c|cuue)

c_deleteData	t_deleteData	cds_deleteData
cd_deleteData	c_substringData	t_substringData
cds_substringData	cd_substringData	

signature p5t5 (c|csue)

di_createDocFromURI

signature p5t6 (c|cuce)

c_insertData	t_insertData	cds_insertData
cd_insertData		

signature p5t7 (c|ccbe)

doc_importNode

signature p6t1 (c|cccce)

el_setAttributeNS	di_createDocumentType	di_createDocument

signature p6t2 (b|ccsue)

di_saveDocToFile

signature p6t3 (c|cuuce)

c_replaceData	t_replaceData	cds_replaceData
cd_replaceData		

In alphabetical order

a_appendChild	(p4t1 c|cce)
a_attributes	(p3t1 c|ce)
a_canAppend	(p4t3 b|cce)
a_childNodes	(p3t1 c|ce)
a_cloneNode	(p4t4 c|cbe)
a_firstChild	(p3t1 c|ce)
a_hasAttributes	(p3t4 b|ce)
a_hasChildNodes	(p3t4 b|ce)
a_insertBefore	(p5t1 c|ccce)
a_lastChild	(p3t1 c|ce)
a_localName	(p3t1 c|ce)
a_name	(p3t1 c|ce)
a_namespaceURI	(p3t1 c|ce)
a_nextSibling	(p3t1 c|ce)
a_nodeName	(p3t1 c|ce)
a_nodeType	(p3t3 u|ce)
a_nodeValue	(p3t1 c|ce)
a_normalize	(p3t1 c|ce)
a_ownerDocument	(p3t1 c|ce)
a_ownerElement	(p3t1 c|ce)
a_parentNode	(p3t1 c|ce)
a_prefix	(p3t1 c|ce)
a_previousSibling	(p3t1 c|ce)
a_removeChild	(p4t1 c|cce)
a_replaceChild	(p5t1 c|ccce)
a_set_nodeValue	(p4t1 c|cce)
a_set_prefix	(p4t1 c|cce)
a_set_value	(p4t1 c|cce)
a_specified	(p3t4 b|ce)
a_unref	(p3t1 c|ce)
a_value	(p3t1 c|ce)
c_appendChild	(p4t1 c|cce)
c_appendData	(p4t1 c|cce)
c_attributes	(p3t1 c|ce)
c_canAppend	(p4t3 b|cce)
c_childNodes	(p3t1 c|ce)
c_cloneNode	(p4t4 c|cbe)
c_data	(p3t1 c|ce)
c_deleteData	(p5t4 c|cuue)
c_firstChild	(p3t1 c|ce)
c_hasAttributes	(p3t4 b|ce)
c_hasChildNodes	(p3t4 b|ce)
c_insertBefore	(p5t1 c|ccce)
c_insertData	(p5t6 c|cuce)
c_lastChild	(p3t1 c|ce)
c_length	(p3t3 u|ce)
c_localName	(p3t1 c|ce)
c_namespaceURI	(p3t1 c|ce)
c_nextSibling	(p3t1 c|ce)
c_nodeName	(p3t1 c|ce)
c_nodeType	(p3t3 u|ce)
c_nodeValue	(p3t1 c|ce)
c_normalize	(p3t1 c|ce)
c_ownerDocument	(p3t1 c|ce)
c_parentNode	(p3t1 c|ce)

c_prefix	(p3t1 c|ce)
c_previousSibling	(p3t1 c|ce)
c_removeChild	(p4t1 c|cce)
c_replaceChild	(p5t1 c|ccce)
c_replaceData	(p6t3 c|cuuce)
c_set_data	(p4t1 c|cce)
c_set_nodeValue	(p4t1 c|cce)
c_set_prefix	(p4t1 c|cce)
c_substringData	(p5t4 c|cuue)
c_unref	(p3t1 c|ce)
cd_appendChild	(p4t1 c|cce)
cd_appendData	(p4t1 c|cce)
cd_attributes	(p3t1 c|ce)
cd_canAppend	(p4t3 b|cce)
cd_childNodes	(p3t1 c|ce)
cd_cloneNode	(p4t4 c|cbe)
cd_data	(p3t1 c|ce)
cd_deleteData	(p5t4 c|cuue)
cd_firstChild	(p3t1 c|ce)
cd_hasAttributes	(p3t4 b|ce)
cd_hasChildNodes	(p3t4 b|ce)
cd_insertBefore	(p5t1 c|ccce)
cd_insertData	(p5t6 c|cuce)
cd_lastChild	(p3t1 c|ce)
cd_length	(p3t3 u|ce)
cd_localName	(p3t1 c|ce)
cd_namespaceURI	(p3t1 c|ce)
cd_nextSibling	(p3t1 c|ce)
cd_nodeName	(p3t1 c|ce)
cd_nodeType	(p3t3 u|ce)
cd_nodeValue	(p3t1 c|ce)
cd_normalize	(p3t1 c|ce)
cd_ownerDocument	(p3t1 c|ce)
cd_parentNode	(p3t1 c|ce)
cd_prefix	(p3t1 c|ce)
cd_previousSibling	(p3t1 c|ce)
cd_removeChild	(p4t1 c|cce)
cd_replaceChild	(p5t1 c|ccce)
cd_replaceData	(p6t3 c|cuuce)
cd_set_data	(p4t1 c|cce)
cd_set_nodeValue	(p4t1 c|cce)
cd_set_prefix	(p4t1 c|cce)
cd_substringData	(p5t4 c|cuue)
cd_unref	(p3t1 c|ce)
cds_appendChild	(p4t1 c|cce)
cds_appendData	(p4t1 c|cce)
cds_attributes	(p3t1 c|ce)
cds_canAppend	(p4t3 b|cce)
cds_childNodes	(p3t1 c|ce)
cds_cloneNode	(p4t4 c|cbe)
cds_data	(p3t1 c|ce)
cds_deleteData	(p5t4 c|cuue)
cds_firstChild	(p3t1 c|ce)
cds_hasAttributes	(p3t4 b|ce)
cds_hasChildNodes	(p3t4 b|ce)
cds_insertBefore	(p5t1 c|ccce)
cds_insertData	(p5t6 c|cuce)
cds_lastChild	(p3t1 c|ce)

cds_length	(p3t3 u|ce)
cds_localName	(p3t1 c|ce)
cds_namespaceURI	(p3t1 c|ce)
cds_nextSibling	(p3t1 c|ce)
cds_nodeName	(p3t1 c|ce)
cds_nodeType	(p3t3 u|ce)
cds_nodeValue	(p3t1 c|ce)
cds_normalize	(p3t1 c|ce)
cds_ownerDocument	(p3t1 c|ce)
cds_parentNode	(p3t1 c|ce)
cds_prefix	(p3t1 c|ce)
cds_previousSibling	(p3t1 c|ce)
cds_removeChild	(p4t1 c|cce)
cds_replaceChild	(p5t1 c|ccce)
cds_replaceData	(p6t3 c|cuuce)
cds_set_data	(p4t1 c|cce)
cds_set_nodeValue	(p4t1 c|cce)
cds_set_prefix	(p4t1 c|cce)
cds_splitText	(p4t2 c|cue)
cds_substringData	(p5t4 c|cuue)
cds_unref	(p3t1 c|ce)
df_appendChild	(p4t1 c|cce)
df_attributes	(p3t1 c|ce)
df_canAppend	(p4t3 b|cce)
df_childNodes	(p3t1 c|ce)
df_cloneNode	(p4t4 c|cbe)
df_firstChild	(p3t1 c|ce)
df_hasAttributes	(p3t4 b|ce)
df_hasChildNodes	(p3t4 b|ce)
df_insertBefore	(p5t1 c|ccce)
df_lastChild	(p3t1 c|ce)
df_localName	(p3t1 c|ce)
df_namespaceURI	(p3t1 c|ce)
df_nextSibling	(p3t1 c|ce)
df_nodeName	(p3t1 c|ce)
df_nodeType	(p3t3 u|ce)
df_nodeValue	(p3t1 c|ce)
df_normalize	(p3t1 c|ce)
df_ownerDocument	(p3t1 c|ce)
df_parentNode	(p3t1 c|ce)
df_prefix	(p3t1 c|ce)
df_previousSibling	(p3t1 c|ce)
df_removeChild	(p4t1 c|cce)
df_replaceChild	(p5t1 c|ccce)
df_set_nodeValue	(p4t1 c|cce)
df_set_prefix	(p4t1 c|cce)
di_createDocFromURI	(p5t5 c|csue)
di_createDocument	(p6t1 c|cccce)
di_createDocumentType	(p6t1 c|cccce)
di_mkref	(p2t1 c|e)
di_saveDocToFile	(p6t2 b|ccsue)
di_unref	(p3t1 c|ce)
doc_appendChild	(p4t1 c|cce)
doc_attributes	(p3t1 c|ce)
doc_canAppend	(p4t3 b|cce)
doc_childNodes	(p3t1 c|ce)
doc_cloneNode	(p4t4 c|cbe)
doc_createAttribute	(p4t1 c|cce)

doc_createAttributeNS	(p5t1 c|ccce)
doc_createCDATASection	(p4t1 c|cce)
doc_createComment	(p4t1 c|cce)
doc_createDocumentFragment	(p3t1 c|ce)
doc_createElement	(p4t1 c|cce)
doc_createElementNS	(p5t1 c|ccce)
doc_createEntityReference	(p4t1 c|cce)
doc_createProcessingInstruction	(p5t1 c|ccce)
doc_createTextNode	(p4t1 c|cce)
doc_doctype	(p3t1 c|ce)
doc_documentElement	(p3t1 c|ce)
doc_firstChild	(p3t1 c|ce)
doc_getElementById	(p4t1 c|cce)
doc_getElementsByTagName	(p4t1 c|cce)
doc_getElementsByTagNameNS	(p5t1 c|ccce)
doc_hasAttributes	(p3t4 b|ce)
doc_hasChildNodes	(p3t4 b|ce)
doc_implementation	(p3t1 c|ce)
doc_importNode	(p5t7 c|ccbe)
doc_insertBefore	(p5t1 c|ccce)
doc_lastChild	(p3t1 c|ce)
doc_localName	(p3t1 c|ce)
doc_namespaceURI	(p3t1 c|ce)
doc_nextSibling	(p3t1 c|ce)
doc_nodeName	(p3t1 c|ce)
doc_nodeType	(p3t3 u|ce)
doc_nodeValue	(p3t1 c|ce)
doc_normalize	(p3t1 c|ce)
doc_ownerDocument	(p3t1 c|ce)
doc_parentNode	(p3t1 c|ce)
doc_prefix	(p3t1 c|ce)
doc_previousSibling	(p3t1 c|ce)
doc_removeChild	(p4t1 c|cce)
doc_replaceChild	(p5t1 c|ccce)
doc_set_nodeValue	(p4t1 c|cce)
doc_set_prefix	(p4t1 c|cce)
doc_unref	(p3t1 c|ce)
dt_appendChild	(p4t1 c|cce)
dt_attributes	(p3t1 c|ce)
dt_canAppend	(p4t3 b|cce)
dt_childNodes	(p3t1 c|ce)
dt_cloneNode	(p4t4 c|cbe)
dt_entities	(p3t1 c|ce)
dt_firstChild	(p3t1 c|ce)
dt_hasAttributes	(p3t4 b|ce)
dt_hasChildNodes	(p3t4 b|ce)
dt_insertBefore	(p5t1 c|ccce)
dt_internalSubset	(p3t1 c|ce)
dt_lastChild	(p3t1 c|ce)
dt_localName	(p3t1 c|ce)
dt_name	(p3t1 c|ce)
dt_namespaceURI	(p3t1 c|ce)
dt_nextSibling	(p3t1 c|ce)
dt_nodeName	(p3t1 c|ce)
dt_nodeType	(p3t3 u|ce)
dt_nodeValue	(p3t1 c|ce)
dt_normalize	(p3t1 c|ce)
dt_notations	(p3t1 c|ce)

dt_ownerDocument	(p3t1 c|ce)
dt_parentNode	(p3t1 c|ce)
dt_prefix	(p3t1 c|ce)
dt_previousSibling	(p3t1 c|ce)
dt_publicId	(p3t1 c|ce)
dt_removeChild	(p4t1 c|cce)
dt_replaceChild	(p5t1 c|ccce)
dt_set_nodeValue	(p4t1 c|cce)
dt_set_prefix	(p4t1 c|cce)
dt_systemId	(p3t1 c|ce)
dt_unref	(p3t1 c|ce)
el_appendChild	(p4t1 c|cce)
el_attributes	(p3t1 c|ce)
el_canAppend	(p4t3 b|cce)
el_childNodes	(p3t1 c|ce)
el_cloneNode	(p4t4 c|cbe)
el_firstChild	(p3t1 c|ce)
el_getAttribute	(p4t1 c|cce)
el_getAttributeNS	(p5t1 c|ccce)
el_getAttributeNode	(p4t1 c|cce)
el_getAttributeNodeNS	(p5t1 c|ccce)
el_getElementsByTagName	(p4t1 c|cce)
el_getElementsByTagNameNS	(p5t1 c|ccce)
el_hasAttributeNS	(p5t3 b|ccce)
el_hasAttributes	(p3t4 b|ce)
el_hasChildNodes	(p3t4 b|ce)
el_insertBefore	(p5t1 c|ccce)
el_lastChild	(p3t1 c|ce)
el_localName	(p3t1 c|ce)
el_namespaceURI	(p3t1 c|ce)
el_nextSibling	(p3t1 c|ce)
el_nodeName	(p3t1 c|ce)
el_nodeType	(p3t3 u|ce)
el_nodeValue	(p3t1 c|ce)
el_normalize	(p3t1 c|ce)
el_ownerDocument	(p3t1 c|ce)
el_parentNode	(p3t1 c|ce)
el_prefix	(p3t1 c|ce)
el_previousSibling	(p3t1 c|ce)
el_removeAttribute	(p4t1 c|cce)
el_removeAttributeNS	(p5t1 c|ccce)
el_removeAttributeNode	(p4t1 c|cce)
el_removeChild	(p4t1 c|cce)
el_replaceChild	(p5t1 c|ccce)
el_setAttribute	(p5t1 c|ccce)
el_setAttributeNS	(p6t1 c|cccce)
el_setAttributeNode	(p4t1 c|cce)
el_setAttributeNodeNS	(p4t1 c|cce)
el_set_nodeValue	(p4t1 c|cce)
el_set_prefix	(p4t1 c|cce)
el_tagName	(p3t1 c|ce)
el_unref	(p3t1 c|ce)
ent_appendChild	(p4t1 c|cce)
ent_attributes	(p3t1 c|ce)
ent_canAppend	(p4t3 b|cce)
ent_childNodes	(p3t1 c|ce)
ent_cloneNode	(p4t4 c|cbe)
ent_firstChild	(p3t1 c|ce)

ent_hasAttributes	(p3t4 b|ce)
ent_hasChildNodes	(p3t4 b|ce)
ent_insertBefore	(p5t1 c|ccce)
ent_lastChild	(p3t1 c|ce)
ent_localName	(p3t1 c|ce)
ent_namespaceURI	(p3t1 c|ce)
ent_nextSibling	(p3t1 c|ce)
ent_nodeName	(p3t1 c|ce)
ent_nodeType	(p3t3 u|ce)
ent_nodeValue	(p3t1 c|ce)
ent_normalize	(p3t1 c|ce)
ent_notationName	(p3t1 c|ce)
ent_ownerDocument	(p3t1 c|ce)
ent_parentNode	(p3t1 c|ce)
ent_prefix	(p3t1 c|ce)
ent_previousSibling	(p3t1 c|ce)
ent_publicId	(p3t1 c|ce)
ent_removeChild	(p4t1 c|cce)
ent_replaceChild	(p5t1 c|ccce)
ent_set_nodeValue	(p4t1 c|cce)
ent_set_prefix	(p4t1 c|cce)
ent_systemId	(p3t1 c|ce)
ent_unref	(p3t1 c|ce)
er_appendChild	(p4t1 c|cce)
er_attributes	(p3t1 c|ce)
er_canAppend	(p4t3 b|cce)
er_childNodes	(p3t1 c|ce)
er_cloneNode	(p4t4 c|cbe)
er_firstChild	(p3t1 c|ce)
er_hasAttributes	(p3t4 b|ce)
er_hasChildNodes	(p3t4 b|ce)
er_insertBefore	(p5t1 c|ccce)
er_lastChild	(p3t1 c|ce)
er_localName	(p3t1 c|ce)
er_namespaceURI	(p3t1 c|ce)
er_nextSibling	(p3t1 c|ce)
er_nodeName	(p3t1 c|ce)
er_nodeType	(p3t3 u|ce)
er_nodeValue	(p3t1 c|ce)
er_normalize	(p3t1 c|ce)
er_ownerDocument	(p3t1 c|ce)
er_parentNode	(p3t1 c|ce)
er_prefix	(p3t1 c|ce)
er_previousSibling	(p3t1 c|ce)
er_removeChild	(p4t1 c|cce)
er_replaceChild	(p5t1 c|ccce)
er_set_nodeValue	(p4t1 c|cce)
er_set_prefix	(p4t1 c|cce)
er_unref	(p3t1 c|ce)
n_appendChild	(p4t1 c|cce)
n_attributes	(p3t1 c|ce)
n_canAppend	(p4t3 b|cce)
n_childNodes	(p3t1 c|ce)
n_cloneNode	(p4t4 c|cbe)
n_firstChild	(p3t1 c|ce)
n_hasAttributes	(p3t4 b|ce)
n_hasChildNodes	(p3t4 b|ce)
n_insertBefore	(p5t1 c|ccce)

n_lastChild	(p3t1 c|ce)
n_localName	(p3t1 c|ce)
n_namespaceURI	(p3t1 c|ce)
n_nextSibling	(p3t1 c|ce)
n_nodeName	(p3t1 c|ce)
n_nodeType	(p3t3 u|ce)
n_nodeValue	(p3t1 c|ce)
n_normalize	(p3t1 c|ce)
n_ownerDocument	(p3t1 c|ce)
n_parentNode	(p3t1 c|ce)
n_prefix	(p3t1 c|ce)
n_previousSibling	(p3t1 c|ce)
n_removeChild	(p4t1 c|cce)
n_replaceChild	(p5t1 c|ccce)
n_set_nodeValue	(p4t1 c|cce)
n_set_prefix	(p4t1 c|cce)
n_unref	(p3t1 c|ce)
nl_item	(p4t2 c|cue)
nl_length	(p3t3 u|ce)
nl_unref	(p3t1 c|ce)
nnm_getNamedItem	(p4t1 c|cce)
nnm_getNamedItemNS	(p5t1 c|ccce)
nnm_item	(p4t2 c|cue)
nnm_length	(p3t3 u|ce)
nnm_removeNamedItem	(p4t1 c|cce)
nnm_removeNamedItemNS	(p5t1 c|ccce)
nnm_setNamedItem	(p4t1 c|cce)
nnm_setNamedItemNS	(p4t1 c|cce)
nnm_unref	(p3t1 c|ce)
not_appendChild	(p4t1 c|cce)
not_attributes	(p3t1 c|ce)
not_canAppend	(p4t3 b|cce)
not_childNodes	(p3t1 c|ce)
not_cloneNode	(p4t4 c|cbe)
not_firstChild	(p3t1 c|ce)
not_hasAttributes	(p3t4 b|ce)
not_hasChildNodes	(p3t4 b|ce)
not_insertBefore	(p5t1 c|ccce)
not_lastChild	(p3t1 c|ce)
not_localName	(p3t1 c|ce)
not_namespaceURI	(p3t1 c|ce)
not_nextSibling	(p3t1 c|ce)
not_nodeName	(p3t1 c|ce)
not_nodeType	(p3t3 u|ce)
not_nodeValue	(p3t1 c|ce)
not_normalize	(p3t1 c|ce)
not_ownerDocument	(p3t1 c|ce)
not_parentNode	(p3t1 c|ce)
not_prefix	(p3t1 c|ce)
not_previousSibling	(p3t1 c|ce)
not_publicId	(p3t1 c|ce)
not_removeChild	(p4t1 c|cce)
not_replaceChild	(p5t1 c|ccce)
not_set_nodeValue	(p4t1 c|cce)
not_set_prefix	(p4t1 c|cce)
not_systemId	(p3t1 c|ce)
not_unref	(p3t1 c|ce)
pi_appendChild	(p4t1 c|cce)

pi_attributes	(p3t1 c|ce)
pi_canAppend	(p4t3 b|cce)
pi_childNodes	(p3t1 c|ce)
pi_cloneNode	(p4t4 c|cbe)
pi_data	(p3t1 c|ce)
pi_firstChild	(p3t1 c|ce)
pi_hasAttributes	(p3t4 b|ce)
pi_hasChildNodes	(p3t4 b|ce)
pi_insertBefore	(p5t1 c|ccce)
pi_lastChild	(p3t1 c|ce)
pi_localName	(p3t1 c|ce)
pi_namespaceURI	(p3t1 c|ce)
pi_nextSibling	(p3t1 c|ce)
pi_nodeName	(p3t1 c|ce)
pi_nodeType	(p3t3 u|ce)
pi_nodeValue	(p3t1 c|ce)
pi_normalize	(p3t1 c|ce)
pi_ownerDocument	(p3t1 c|ce)
pi_parentNode	(p3t1 c|ce)
pi_prefix	(p3t1 c|ce)
pi_previousSibling	(p3t1 c|ce)
pi_removeChild	(p4t1 c|cce)
pi_replaceChild	(p5t1 c|ccce)
pi_set_data	(p4t1 c|cce)
pi_set_nodeValue	(p4t1 c|cce)
pi_set_prefix	(p4t1 c|cce)
pi_target	(p3t1 c|ce)
pi_unref	(p3t1 c|ce)
str_len	(p3t3 u|ce)
str_length	(p3t3 u|ce)
str_mkref	(p3t2 c|se)
str_new	(p3t2 c|se)
str_toFortran	(p5t2 b|csue)
str_unref	(p3t1 c|ce)
t_appendChild	(p4t1 c|cce)
t_appendData	(p4t1 c|cce)
t_attributes	(p3t1 c|ce)
t_canAppend	(p4t3 b|cce)
t_childNodes	(p3t1 c|ce)
t_cloneNode	(p4t4 c|cbe)
t_data	(p3t1 c|ce)
t_deleteData	(p5t4 c|cuue)
t_firstChild	(p3t1 c|ce)
t_hasAttributes	(p3t4 b|ce)
t_hasChildNodes	(p3t4 b|ce)
t_insertBefore	(p5t1 c|ccce)
t_insertData	(p5t6 c|cuce)
t_lastChild	(p3t1 c|ce)
t_length	(p3t3 u|ce)
t_localName	(p3t1 c|ce)
t_namespaceURI	(p3t1 c|ce)
t_nextSibling	(p3t1 c|ce)
t_nodeName	(p3t1 c|ce)
t_nodeType	(p3t3 u|ce)
t_nodeValue	(p3t1 c|ce)
t_normalize	(p3t1 c|ce)
t_ownerDocument	(p3t1 c|ce)
t_parentNode	(p3t1 c|ce)

t_prefix	(p3t1 c|ce)
t_previousSibling	(p3t1 c|ce)
t_removeChild	(p4t1 c|cce)
t_replaceChild	(p5t1 c|ccce)
t_replaceData	(p6t3 c|cuuce)
t_set_data	(p4t1 c|cce)
t_set_nodeValue	(p4t1 c|cce)
t_set_prefix	(p4t1 c|cce)
t_splitText	(p4t2 c|cue)
t_substringData	(p5t4 c|cuue)
t_unref	(p3t1 c|ce)