

# XcodeML/Fortran Specification

Version 1.0

Omni Compiler Project

July, 2017

Copyright ©2008-2017 Omni Compiler Project (RIKEN AICS), Permission to copy without fee all or part of this material is granted, provided the XcodeML/Fortran Specification Working Group copyright notice and the title of this document appear. Notice is given that copying is by permission of Omni Compiler Project (RIKEN AICS).

# History

## Version 1.0: July, 2017.

- Supported the Fortran2008 features.
- Corrected the descriptions on the current specification.
  - Corrected the description on the I/O specifier.  
(wrong) value attribute  
(correct) child element
  - Added the missed attribute `is_assumed_shape` to the `indexRange` element.

## Version 0.9J: March 31, 2011

- Added the `FassignStatement` element.
- Added the `FpointerAssignStatement` element.
- Deleted the `assignExpr` element.
- Deleted the `FpointerAssignExpr` element.
- Deleted the `assignExpr` element and `FpointerAssignExpr` elements in the `exprModel` model.
- Added the `FassignStatement` and `FpointerAssignStatement` elements to the `statementModel` model.
- Added the `is_assumed_size` attribute to the `indexRange` element.
- Added the `is_intrinsic` attribute to the `functionCall` element.
- Updated the description on the `statementLabel` element.
- Support for `coarray`.
  - Modified the `FbasicType` element.
  - Added the `coShape` element.
  - Added the `FcoArrayRef` element.
  - Modified the `varRef` element.
  - Modified the `alloc` element.
  - Added the `FcoArrayRef` element to the `exprModel` model.

# Contents

<b>1</b>	<b>Introduction</b>	<b>iii</b>
<b>2</b>	<b>XcodeProgram element</b>	<b>iii</b>
<b>3</b>	<b>Type Table element</b>	<b>iv</b>
3.1	typeTable element . . . . .	iv
3.2	coShape element . . . . .	iv
3.3	FbasicType element . . . . .	iv
3.4	FfunctionType element . . . . .	viii
3.5	FstructType element . . . . .	ix
3.6	typeParams element . . . . .	xi
3.7	typeParam element . . . . .	xi
3.8	typeParamValues element . . . . .	xi
3.9	typeBoundProcedures element . . . . .	xii
3.10	typeBoundProcedure element . . . . .	xii
3.11	typeBoundGenericProcedure element . . . . .	xiii
3.12	finalProcedure element . . . . .	xiii
3.13	binding element . . . . .	xiv
3.14	FenumType element . . . . .	xiv
<b>4</b>	<b>Symbol list</b>	<b>xiv</b>
4.1	globalSymbols element . . . . .	xiv
4.2	symbols element . . . . .	xv
<b>5</b>	<b>Definition and declaration element</b>	<b>xv</b>
5.1	globalDeclarations element . . . . .	xv
5.2	declarations element . . . . .	xv
5.3	FfunctionDefinition element . . . . .	xvi
5.4	varDecl element . . . . .	xvi
5.5	FstructDecl element . . . . .	xvii
5.6	externDecl element . . . . .	xvii
5.7	FmoduleDefinition element . . . . .	xvii
5.8	FuseDecl element . . . . .	xviii
5.9	FuseOnlyDecl element . . . . .	xviii
5.10	FinterfaceDecl element . . . . .	xix
5.11	FmoduleProcedureDecl element . . . . .	xix
5.12	FfunctionDecl element . . . . .	xx
5.13	FprocedureDecl element . . . . .	xx
5.14	FimportDecl element . . . . .	xxi
<b>6</b>	<b>Statement element</b>	<b>xxi</b>
6.1	FassignStatement element . . . . .	xxi
6.2	exprStatement element . . . . .	xxi
6.3	FpointerAssignStatement element . . . . .	xxii
6.4	FifStatement element . . . . .	xxii
6.5	FdoStatement element . . . . .	xxiii
6.6	FdoWhileStatement element . . . . .	xxiii
6.7	continueStatement element . . . . .	xxiii
6.8	FcycleStatement element . . . . .	xxiv

6.9	FexitStatement element	xxiv
6.10	FreturnStatement element	xxiv
6.11	gotoStatement element	xxv
6.12	statementLabel element	xxv
6.13	FselectCaseStatement element	xxvi
6.14	FcaseLabel element	xxvi
6.15	FwhereStatement element	xxvii
6.16	FstopStatement element	xxvii
6.17	Input/Output element	xxvii
6.17.1	FreadStatement, FwriteStatement element	xxvii
6.17.2	FprintStatement element	xxviii
6.17.3	FrewindStatement, FendFileStatement, FbackspaceStatement element	xxix
6.17.4	FopenStatement element	xxix
6.17.5	FcloseStatement element	xxx
6.17.6	FinquireStatement element	xxxi
6.17.7	FwaitStatement element	xxxii
6.17.8	FflushStatement element	xxxiii
6.18	FformatDecl element	xxxiv
6.19	FdataDecl element	xxxiv
6.20	FnamelistDecl element	xxxv
6.21	FequivalenceDecl element	xxxv
6.22	FcommonDecl element	xxxv
6.23	FentryDecl element	xxxvi
6.24	Allocation element	xxxvi
6.24.1	FallocateStatement element	xxxvi
6.24.2	FdeallocateStatement element	xxxvii
6.24.3	FnullifyStatement element	xxxviii
6.25	FpragmaStatement element	xxxviii
6.26	FcontainsStatement element	xxxviii
6.27	condition element	xxxix
6.28	then element	xxxix
6.29	else element	xxxix
6.30	alloc element	xl
6.31	allocOpt element	xl
6.32	forallStatement element	xli
6.33	FdoConcurrentStatement element	xlii
6.34	selectTypeStatement element	xliii
6.35	typeGuard element	xliv
6.36	syncAllStatement element	xlv
6.37	syncImagesStatement element	xlv
6.38	syncMemoryStatement element	xlv
6.39	lock/unlockStatement element	xlvi
6.40	syncStat element	xlvi
6.41	criticalStatement element	xlvii
6.42	associateStatement element	xlvii
6.43	blockStatement element	xlviii

<b>7</b>	<b>Expression element</b>	<b>xlix</b>
7.1	Constant element . . . . .	xlix
7.1.1	FintConstant, FrealConstant, FcharacterConstant, FlogicalConstant element . . . . .	xlix
7.1.2	FcomplexContant element . . . . .	l
7.2	Array constructor element . . . . .	l
7.2.1	FarrayConstructor element . . . . .	l
7.3	Structure constructor element . . . . .	li
7.3.1	Constant element . . . . .	li
7.4	Elements referencing variables . . . . .	lii
7.4.1	Var element . . . . .	lii
7.4.2	FmemberRef element . . . . .	lii
7.4.3	FcoArrayRef element . . . . .	liii
7.4.4	FarrayRef element . . . . .	liii
7.4.5	FcharacterRef element . . . . .	liv
7.4.6	varRef element . . . . .	liv
7.5	Function call . . . . .	lv
7.5.1	functionCall element . . . . .	lv
7.5.2	arguments element . . . . .	lv
7.6	Binary operation element . . . . .	lvi
7.7	Unary operation element . . . . .	lvii
<b>8</b>	<b>General element</b>	<b>lvii</b>
8.1	kind element . . . . .	lvii
8.2	id element . . . . .	lvii
8.3	name element . . . . .	lviii
8.4	value element . . . . .	lviii
8.5	kind params . . . . .	lix
8.6	len element . . . . .	lix
8.7	body element . . . . .	lx
8.8	rename element . . . . .	lx
8.9	renamable element . . . . .	lx
8.10	arrayIndex element . . . . .	lxi
8.11	indexRange element . . . . .	lxi
8.12	lowerBound element . . . . .	lxii
8.13	upperBound element . . . . .	lxii
8.14	step element . . . . .	lxii
8.15	FdoLoop element . . . . .	lxiii
8.16	namedValue element . . . . .	lxiii
8.17	namedValueList element . . . . .	lxiv
8.18	valueList element . . . . .	lxiv
8.19	varList element . . . . .	lxiv
<b>9</b>	<b>Common element</b>	<b>lxv</b>
9.1	The data type name identifier . . . . .	lxv
9.2	Common attributes of the definition / declaration / statement element . . . . .	lxv
9.3	lValueModel model . . . . .	lxv
9.4	exprModel model . . . . .	lxv
9.5	statementModel model . . . . .	lxvi

<b>10 Miscellaneous</b>	<b>lxvi</b>
10.1 Intrinsic procesures . . . . .	lxvi
10.2 Treatment of comemts and pragmas . . . . .	lxvi
10.3 Canonicalization . . . . .	lxvi
10.4 Restrictions . . . . .	lxvii
<b>11 Bibliography</b>	<b>lxvii</b>

## List of Figures

## List of Tables



## 1 Introduction

This document describes the specification of XcodeML for Fortran 90, which is defined based on XcodeML that was originally defined as an intermediate code of the computer languages, mainly C. To clearly distinguish both specifications, we call this specification XcodeML/Fortran. The version of XcodeML on which this is based is 0.8J.

XcodeML/Fortran has the following functions and characteristics:

- Preserves information that can be used to reconstruct a Fortran 90 program,
- Can represent the type information of the Fortran 90 programming language,
- Has syntax elements necessary for a variety of transformations, and
- Has a human-readable format (XML).

XcodeML and XcodeML/Fortran are primarily designed to be convenient to use as an intermediate code in the source code to source code converter systems(hereafter called translators). The program which converts code written in a computer language to XcodeML or XcodeML/Fortran is called the frontend program(hereinafter called frontend), the one which converts XcodeML or XcodeML/Fortran to code written in the computer language is called the backend program(hereinafter called backend) or the decompiler. In addition, there are analysis/converter programs which input/output intermediate code to statically analyze or parallelize a program within the source code translator systems.

## 2 XcodeProgram element

Programs written in XcodeML/Fortran comprises a Type table and global external definitions. The top-level element in an XcodeML/Fortran file is the **XcodeProgram** element:

### Contents model

(typeTable, globalSymbols, globalDeclarations)

### Child elements

name	description	R/O
typeTable	information on data type used by the program	R
globalSymbols	information on global variables used by the program	R
globalDeclarations	information about function and variable declarations	R

### Attributes

name	type	description	R/O
compiler-info	text	F-to-F translator information	O
version	text	F-to-F translator version information	O
time	text	Date and time of translation	O
language	text	source language information	O
source	text	source code information	O



## 3 Type Table element

### 3.1 typeTable element

The `typeTable` element defines the data type information for the entire file. The element comprises the data type definition elements. Data type definition elements consists of the following elements:

#### Contents model

(FbasicType | FfunctionType | FstructType | FenumType))\*

#### Child elements

name	description	R/O
FbasicType	the basic data type	O
FfunctionType	the function data type	O
FstructType	the derived data type	O
FenumType	the enumeration data type	O

#### Attributes

name	type	description	R/O
-	-	-	-

### 3.2 coShape element

The `coShape` element defines a coshape of a coarray.

#### Contents model

(indexRange+)

#### Child elements

name	description	R/O
indexRange	the lower/upper bound of codimension	R

#### Attributes

name	type	description	R/O
-	-	-	-

### 3.3 FbasicType element

The `FbasicType` element defines a reference to primitive types or other type definition elements.

#### Contents model

(kind?, (len | (arrayIndex | indexRange)+)?, coShape?, typeParamValues?)

**Child elements**

name	description	R/O
<b>kind</b>	Specifies the kind parameter of the type element.	O
<b>len</b>	Specifies the length of the character string. Can be specified only when the type attribute of the type is <b>Fcharacter</b> .	O
<b>arrayIndex</b>	Specified if the type element is the array type and the size is expressed by the number of the elements. Can be repeated the number of the dimension times together with the <b>indexRange</b> element.	O
<b>indexRange</b>	Specified if the type element is the array type and the size is expressed by the upper and lower bounds. Can be repeated the number of the dimension times together with the <b>arrayIndex</b> element.	O
<b>coShape</b>	Specifies the coshape of the coarray.	O
<b>typeParamValues</b>	Specifies the type parameter values.	O

**Attributes**

name	type	description	R/O
<code>type</code>	text	Specifies the type identifier of the type element in the XcodeML/Fortran.	R
<code>ref</code>	text	Specifies the type identifier the type refers in the XcodeML/Fortran.	O
<code>is_public</code>	bool	true if the type has the <code>public</code> attribute.	O
<code>is_private</code>	bool	true if the type has the <code>private</code> attribute.	O
<code>is_pointer</code>	bool	true if the type has the <code>pointer</code> attribute.	O
<code>is_target</code>	bool	true if the type has the <code>target</code> attribute.	O
<code>is_external</code>	bool	true if the type has the <code>external</code> attribute.	O
<code>is_intrinsic</code>	bool	true if the type has the <code>intrinsic</code> attribute.	O
<code>is_optional</code>	bool	true if the type has the <code>optional</code> attribute.	O
<code>is_save</code>	bool	true if the type has the <code>save</code> attribute.	O
<code>is_parameter</code>	bool	true if the type has the <code>parameter</code> attribute.	O
<code>is_allocatable</code>	bool	true if the type has the <code>allocatable</code> attribute.	O
<code>intent</code>	text	Specifies the intent attribute the value of which is 'in', 'out' or 'inout'.	O
<code>is_protected</code>	bool	true if the type has the <code>protected</code> attribute.	O
<code>is_value</code>	bool	true if the type has the <code>value</code> attribute.	O
<code>is_volatile</code>	bool	true if the type has the <code>volatile</code> attribute.	O
<code>is_asynchronous</code>	bool	true if the type has the <code>asynchronous</code> attribute.	O
<code>is_contiguous</code>	bool	true if the type has the <code>contiguous</code> attribute.	O
<code>is_class</code>	bool	true if the type is the <code>class</code> type. If no type identifier( <code>ref</code> ) is specified, the type represents ' <code>class(*)</code> '.	O
<code>is_procedure</code>	bool	true if the type is the <code>procedure</code> type. If no type identifier( <code>ref</code> ) is specified, the type represents ' <code>procedure()</code> '.	O
<code>pass</code>	text	' <code>nopass</code> ' or ' <code>pass</code> ' if the <code>pass</code> attribute is specified.	O
<code>pass_arg_name</code>	text	If the <code>pass</code> attribute is specified and it has an argument, specifies the argument name.	O
<code>bind</code>	text	Specifies the kind if the type has the <code>bind</code> attribute. Only 'C' is available currently.	O
<code>bind_name</code>	text	Specifies the bind name.	O

**Example**

The data type definition for "`integer(kind=8)`" is as follows:

```

XcodeML/Fortran
<FbasicType type="TYPE_NAME" ref="Fint">
  <kind>8</kind>
</FbasicType>
```

The data type definition for "`integer dimension(10, 1:10)`" is as follows:

```

XcodeML/Fortran
<FbasicType type="TYPE_NAME" ref="Fint">
  <arrayIndex>
    <FintConstant type="Fint">10</FintConstant>
  </arrayIndex>
```

```

5 <indexRange>
  <lowerBound>
    <FintConstant type="Fint">1</FintConstant>
  </lowerBound>
  <upperBound>
10   <FintConstant type="Fint">10</FintConstant>
  </upperBound>
</FbasicType>

```

The data type definition for "character(len=10)" is as follows:

```

XcodeML/Fortran
<FbasicType type="TYPE_NAME" ref="Fcharacter">
  <len>
    <FintConstant type="Fint">10</FintConstant>
  </len>
5 </FbasicType>

```

The data type definition for "type(s(selected\_int\_kind(10),100, :))" is as follows:

```

XcodeML/Fortran
<FbasicType type="TYPE_NAME" ref="TYPE_ID_OF_S">
<typeParamValues>
  <functionCall type="Fint" is_intrinsic="true">
    <name>selected_real_kind</name>
5   <arguments>
    <FintConstant type="Fint">10</FintConstant>
    </arguments>
  </functionCall>
  <FintConstant type="Fint">100</FintConstant>
10   <len></len>
  <typeParamValues>
</FbasicType>

```

The data type definition for "class(ss)" is as follows:

```

XcodeML/Fortran
<FbasicType type="TYPE_NAME" ref="TYPE_ID_OF_S" is_class="true">
</FbasicType>

```

The data type definition for "procedure(func1),pointer" is as follows:

```

XcodeML/Fortran
<FbasicType type="TYPE_NAME" ref="TYPE_ID_OF_FUNC1" is_pointer="true"
  is_procedure="true">
</FbasicType>

```

The data type definition for "integer,bind(c,name="cname")" is as follows:

```

XcodeML/Fortran
<FbasicType type="TYPE_NAME" ref="Fint" is_bind="C" bind_name="cname">
</FbasicType>

```

### 3.4 FfunctionType element

The FfunctionType element defines a function data type.

#### Contents model

(params?)

#### Child elements

name	description	R/O
params	Specifies the names of the dummy arguments if the type element has arguments.	O

#### Attributes

name	type	description	R/O
type	text	Specifies the type identifier of the type element in the XcodeML/Fortran.	R
return_type	text	Specifies the type identifier of the data type the function returns in the XcodeML/Fortran. If the value is 'Fvoid', the type element is the subroutine, otherwise the function.	R
result_name	text	Specifies the name of the variable to set the return value specified by result clause.	O
is_recursive	bool	Specifies if the type element has the recursive attribute.	O
is_program	bool	Specifies if the type element is a main program. 1(or true) is specified if the type element is a main program. 0(or false) is specifies if the type element is the function or subroutine depending on the type of the return_type attribute.	O
is_internal	bool	Specifies if the type element is an internal subprogram.	O
bind	text	Specifies the kind if the type has the bind attribute. Only 'C' is available currently.	O
bind_name	text	Specifies if the bind name.	O

#### Example

The data type of the function foo below is as follows:

```

_____ Fortran 90 _____
function foo(a, b)
  integer a, b
  real foo
```

```

_____ XcodeML/Fortran _____
<FfunctionType type="F0" return_type="Freal">
  <params>
    <name type="Fint">a</name>
    <name type="Fint">b</name>
  </params>
</FfunctionType>
```

### 3.5 FstructType element

The FstructType element defines a derived type.

#### Contents model

(typeParams? symbols? typeBoundProcedures?)

#### Child elements

name	description	R/O
symbols	Specifies the members of the type element.	R

#### Attributes

name	type	description	R/O
type	text	Specifies the type identifier of the type element in the XcodeML/Fortran.	R
is_public	bool	Specifies if the type element has the <code>public</code> attribute as the access qualifier. The default value is 0(or false).	O
is_private	bool	Specifies if the type element has the <code>private</code> attribute as the access qualifier. The default value is 0(or false).	O
is_sequence	bool	Specifies if the type element has the <code>sequence</code> attribute in the type declaration statement. The default value is 0(or false).	O
is_internal_private	bool	Specifies if the type element has the <code>private</code> attribute in the derived type definition. If so, 1(or true) is set. By the restriction of Fortran 90, the <code>is_internal_private</code> attribute cannot be specified if <code>is_sequence</code> is 0(or false).	O
is_abstract	bool	<code>true</code> if the type has the <code>abstract</code> attribute.	O
extends	text	Specifies the type identifier of the parent type if the type is the extended type.	O
bind	text	Specifies the kind if the type has the <code>bind</code> attribute. Only 'C' is available currently.	O

#### Example

The FstructType element for the type S below is as follows:

Fortran 90

```

type S
  integer x1, y1;
end type S

```

XcodeML/Fortran

```

<estructType type="TYPE_NAME">
  <symbols>
    <id type="Fint">
      <name type="Fint">x1</name>
    </id>
    <id type="Fint">
      <name type="Fint">y1</name>

```

```

    </id>
  </symbols>
10 </FstructType>

```

The FstructType element for the type SS below is as follows:

Fortran 2008

```

type SS extend S
  integer,kind:: kind
  integer,length:: n
  real(kind):: a(n);
5 contains
  procedure:: f1=>f1e;
  procedure,pass(arg):: f2=>f2e;
  generic:: ff=>f1,f2
  final:: fn
10 end type

```

XcodeML/Fortran

```

<FstructType extends="TYPE_ID_OF_S">
  <typeParams>
    <typeParam attr="kind">
      <name>kind</name>
5    </typeParam>
    <typeParam attr="len">
      <name>n</name>
    </typeParam>
  </typeParams>
10 <symbols>
  <id>
    <name>a</name>
  </id>
  </symbols>
15 <typeBoundProcedures>
  <typeBoundProcedure>
    <name>f1</name>
    <binding>
      <name>f1e</name>
20    </binding>
  </typeBoundprocedure>
  <typeBoundProcedure pass="pass" arg_name="arg">
    <name>f2</name>
    <binding>
      <name>f2e</name>
25    </binding>
  </typeBoundProcedure>
  <typeBoundGenericProcedure>
    <name>ff</name>
30  <binding>
    <name>f1</name>

```

```

<name>f2</name>
</binding>
  </typeBoundProcedure>
35 <finalProcedure>
    <name>fn</name>
    </finalProcedure>
  </typeBoundProcedures>
</FstructType>

```

### 3.6 typeParams element

The typeParams element defines type parameters of the derived type.

#### Contents model

(typeParam+)

#### Child elements

name	description	R/O
typeParam	Specifies the type parameters.	O

#### Attributes

name	type	description	R/O
-	-	-	-

### 3.7 typeParam element

The typeParam element defines each type parameter.

#### Contents model

(name, value?)

#### Child elements

name	description	R/O
name	Specifies the name of the type parameter.	R
value	Specifies the initial value of the type parameter.	O

#### Attributes

name	type	description	R/O
type	text	Specifies the type identifier in XcodeML/Fortran representing the type element.	R
attr	text	Specifies the parameter attribute by either of "kind" or "length".	R

### 3.8 typeParamValues element

The typeParamValues element defines type parameter values.



**Contents model**

((exprModel | len)+)

**Child elements**

name	description	R/O
exprModel	Specifies the value of the type parameter.	O
len	Specifies the value of the type parameter if the value is assumed('*') or deferred(':').	O

**Attributes**

name	type	description	R/O
-	-	-	-

**3.9 typeBoundProcedures element**

The typeBoundProcedures element defines type-bound procedures.

**Contents model**

((typeBoundprocedure|typeBoundGenericProcedure|finalProcedure)+)

**Child elements**

name	description	R/O
typeBoundProcedure	Specifies the type-bound procedure.	O
typeBoundGenericProcedure	Specifies the generic type-bound procedure.	O
finalProcedure	Specifies the final subroutine.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**3.10 typeBoundProcedure element**

The typeBoundProcedure element defines each type-bound procedure.

**Contents model**

(name, binding?)

**Child elements**

name	description	R/O
name	Specifies the name of the binding.	R
binding	Specifies the procedure name the binding name is bound.	O

**Attributes**

name	type	description	R/O
<code>pass</code>	text	"nopass" or "pass" if the <code>pass</code> attribute is specified.	O
<code>pass_arg_name</code>	text	If the <code>pass</code> attribute is specified and it has an argument, specifies the argument name.	O
<code>is_non_overridable</code>	bool	true if the element has the <code>non_overridable</code> attribute.	O
<code>is_deferred</code>	bool	true if the element has the <code>deferred</code> attribute.	O
<code>is_public</code>	bool	true if the element has the <code>public</code> attribute.	O
<code>is_private</code>	bool	true if the element has the <code>private</code> attribute.	O

**3.11 typeBoundGenericProcedure element**

The `typeBoundGenericProcedure` element defines a generic type-bound procedure.

**Contents model**

(name, binding)

**Child elements**

name	description	R/O
<code>name</code>	Specifies the generic name or the name of the defined operator. Otherwise, empty.	R
<code>binding</code>	Specifies the procedure names the binding name is bound.	R

**Attributes**

name	type	description	R/O
<code>is_operator</code>	bool	Specifies if the element represents a defined operator.	O
<code>is_assignment</code>	bool	Specifies if the element represents an assignment operator.	O
<code>is_defined_io</code>	text	Either of "read(formatted)", "read(unformatted)", "write(formatted)" or "write(unformatted)" is specified if the element represents a user-defined I/O procedure.	O
<code>is_public</code>	bool	true if the element has the <code>public</code> attribute.	O
<code>is_private</code>	bool	true if the element has the <code>private</code> attribute.	O

**3.12 finalProcedure element**

The `finalProcedure` element defines a `final` subroutine.

**Contents model**

(name)

**Child elements**

name	description	R/O
<code>name</code>	Specifies the name of the <code>final</code> subroutine.	R

**Attributes**

name	type	description	R/O
-	-	-	-

**3.13 binding element**

The binding element represents a procedure list to bind.

**Contents model**

(name+)

**Child elements**

name	description	R/O
name	Specifies the procedure name to bind, the interface name or the bound name.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**3.14 FenumType element**

The FenumType element defines a enumeration data type.

**Contents model**

((name, value?)+)

**Child elements**

name	description	R/O
name	Specifies the name of the enumerator.	R
value	Specifies the value of the enumerator.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**4 Symbol list****4.1 globalSymbols element**

The globalSymbols element defines identifiers having the global scope.

**Contents model**

(id\*)

**Child elements**

name	description	R/O
id	Specifies the identifier having the global scope.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**4.2 symbols element**

The `symbols` element defines identifiers having the local scope.

**Contents model**

(`id`\*)

**Child elements**

name	description	R/O
<code>id</code>	Specifies the identifier having the local scope.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**5 Definition and declaration element****5.1 globalDeclarations element**

The `globalDeclarations` element is the element to declare external variables and to define functions in the program.

**Contents model**

(`FfunctionDefinition` | `FmoduleDefinition`)\*

**Child elements**

name	description	R/O
<code>FfunctionDefinition</code>	the definition of a Fortran 90 main program, function or subroutine.	O
<code>FmoduleDefinition</code>	the definition of a Fortran 90 module.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**5.2 declarations element**

The `declarations` element is the element to declare variables etc. in the program.

**Contents model**

(`varDecl` | `FstructDecl` | `externDecl` | `FuseDecl` | `FuseOnlyDecl` | `FinterfaceDecl` | `FnamelistDecl` | `FequivalenceDecl` | `FcommonDecl` | `FprocedureDecl` | `FimportDecl`)\*

**Child elements**

name	description	R/O
varDecl	the definition of a variable	O
FstructDecl	the Fortran 90 derived type definition	O
externDecl	the definition of an external variable or an identifier	O
FuseDecl	the Fortran 90 use declaration	O
FuseOnlyDecl	the Fortran 90 use declaration with the <b>only</b> option	O
FinterfaceDecl	the Fortran 90 <b>interface</b> statement	O
FnamelistDecl	the Fortran 90 <b>namelist</b> statement	O
FequivalenceDecl	the Fortran 90 <b>equivalence</b> statement	O
FcommonDecl	the Fortran 90 <b>common</b> statement	O
FprocedureDecl	the Fortran 2003 <b>procedure</b> statement	O
FimportDecl	the Fortran 2003 <b>import</b> statement	O

**Attributes**

name	type	description	R/O
-	-	-	-

**5.3 FfunctionDefinition element**

The `FfunctionDefinition` element defines a program, function or subroutine. Differently from C language, arguments in Fortran are passed by references, so we prepare another entry than `functionDefinition` of XcodeML.

**Contents model**

(name, symbols?, params?, declarations?, body)

**Child elements**

name	description	R/O
name	Specifies the name of the function or the subroutine.	R
symbols	Specifies the symbols included in the element.	O
params	Specifies the dummy arguments.	O
declarations	Specifies the definitions and declarations included in the element.	O
body	Specifies the executable statements included in the element.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**5.4 varDecl element**

The `varDecl` element declares a variable.

**Contents model**

(name, value?)

**Child elements**

name	description	R/O
name	Specifies the name of the variable.	R
value	Specifies the initial value if present.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**5.5 FstructDecl element**

The `FstructDecl` element defines the derived type.

**Contents model**

(name)

**Child elements**

name	description	R/O
name	Specifies the name of the derived type.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**5.6 externDecl element**

The `externDecl` element declares an external definition.

**Contents model**

(name)

**Child elements**

name	description	R/O
name	Specifies the name of the identifier of the external definition to declare.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**5.7 FmoduleDefinition element**

The `FmoduleDefinition` element represents a module statement.

**Contents model**

(symbols?, declarations?, FcontainsStatement?)

**Child elements**

name	description	R/O
symbols	Specifies the symbols included in the element.	O
declarations	Specifies the definitions and declarations included in the element.	O
FcontainsStatement	Specifies the <code>contain</code> statement.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
name	text	Specifies the module name.	R
is_sub	bool	<code>true</code> if the element represents a submodule.	O
parent_name	text	Specifies the name of the parent module.	O

**5.8 FuseDecl element**The FuseDecl element represents an `use` statement without the `only` option.**Contents model**

(rename\*)

**Child elements**

name	description	R/O
rename	Specifies the rename.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
name	text	Specifies the module name.	R
intrinsic	bool	<code>true</code> if the <code>intrinsic</code> attribute is specified. <code>false</code> if the <code>non_intrinsic</code> attribute is specified.	O

**5.9 FuseOnlyDecl element**The FuseOnlyDecl element represents an `use` statement with the `only` option.**Contents model**

(renamable\*)

**Child elements**

name	description	R/O
renamable	Specifies the only list.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
name	text	Specifies the module name.	R
intrinsic	bool	true if the intrinsic attribute is specified. false if the non_intrinsic attribute is specified.	O

**5.10 FinterfaceDecl element**

The FinterfaceDecl element represents an interface statement.

**Contents model**

(FmoduleProcedureDecl | FfunctionDecl)\*

**Child elements**

name	description	R/O
FmoduleProcedureDecl	Specifies the module procedure statements included in the element.	O
FfunctionDecl	Specifies the functions and subroutines included in the element.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
name	text	Specifies the generic name or the defined operator name.	O
is_operator	bool	Specifies if the interface is a defined operator.	O
is_assignment	bool	Specifies if the interface is a defined assignment operator.	O
is_defined_io	bool	Either of "read(formatted)", "read(unformatted)", "write(formatted)" or "write(unformatted)" is specified if the element represents a user-defined I/O procedure.	O
is_abstract	bool	true if the element represents an abstract interface.	O

**5.11 FmoduleProcedureDecl element**

The FmoduleProcedureDecl represents a module procedure statement in an interface statement.

**Contents model**

(name\*)



**Child elements**

name	description	R/O
name	Specifies the procedure names.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**5.12 FfunctionDecl element**

The FfunctionDecl element represents a **function** or **subroutine** statement in an **interface** statement.

**Contents model**

(name, params?, declarations?)

**Child elements**

name	description	R/O
name	Specifies the name of the function or the subroutine.	R
params	Specifies the dummy arguments.	O
declarations	Specifies the definitions and declarations in the element.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**5.13 FprocedureDecl element**

The FprocedureDecl element represents a **procedure** statement.

**Contents model**

(name, value?)

**Child elements**

name	description	R/O
name	Specifies the name of the procedure.	R
value	Specifies the initial value if present.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

### 5.14 FimportDecl element

The FimportDecl element represents an import statement.

#### Contents model

(name\*)

#### Child elements

name	description	R/O
name	Specifies the name to import.	O

#### Attributes

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

## 6 Statement element

This is an XML element which corresponds to the syntax element in the Fortran 90 statement. The line number is attached to each element as an attribute, which can be used to extract file information and the line number where the particular statement is found.

### 6.1 FassignStatement element

The FassignStatement element represents an assignment statement.

#### Contents model

(lValueModel, exprModel)

#### Child elements

name	description	R/O
lValueModel	Specifies the left-hand side expression. Refer to "9.3 lValue-Model".	R
exprModel	Specifies the right-hand side expression. Refer to "9.4 exprModel".	R

#### Attributes

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

### 6.2 exprStatement element

The exprStatement element represents a statement expressed by an expression.

#### Contents model

(exprModel)

**Child elements**

name	description	R/O
exprModel	Specifies the expression. Refer to "9.4 exprModel"	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.3 FpointerAssignStatement element**

The FpointerAssignStatement element represents a pointer assignment statement.

**Contents model**

(lValueMode, exprModel)

**Child elements**

name	description	R/O
lValueModel	Specifies the left-hand side expression. Refer to "9.3 lValue-Model".	R
exprModel	Specifies the right-hand side expression. Refer to "9.4 exprModel".	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.4 FifStatement element**

The FifStatement element represents an if statement.

**Contents model**

(condition, then, else?)

**Child elements**

name	description	R/O
condition	Specifies the conditional expression.	R
then	Specifies the statement to execute if the condition falls true.	R
else	Specifies the statement to execute if the condition falls false.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
construct_name	text	Specifies the construct name.	O

### 6.5 FdoStatement element

The `FdoStatement` element represents a `do` statement. The labelled `do` construct must be replaced by an equivalent `do` statement with the label eliminated.

#### Contents model

(`Var?`, `indexRange?`, `body?`)

#### Child elements

name	description	R/O
<code>Var</code>	Specifies the index variable.	O
<code>indexRange</code>	Specifies the range of of the value of the index variable.	O
<code>body</code>	Specifies the statements included in the <code>do</code> statement.	O

#### Attributes

name	type	description	R/O
<code>common attriubutes</code>	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
<code>construct_name</code>	text	Specifies the construct name.	O

### 6.6 FdoWhileStatement element

The `FdoWhileStatement` element represents a `do` statement with `while` used as the controlling expression. The labelled `do while` construct must be replaced by an equivalent `do while` statement with the label eliminated.

#### Contents model

(`condition`, `body?`)

#### Child elements

name	description	R/O
<code>condition</code>	Specifies the conditional expression.	R
<code>body</code>	Specifies the statements included in the <code>do while</code> statement.	O

#### Attributes

name	type	description	R/O
<code>common attriubutes</code>	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
<code>construct_name</code>	text	Specifies the construct name.	O

### 6.7 continueStatement element

The `continueStatement` element represents a `continue` statement.

#### Contents model

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
common attriubutes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.8 FcycleStatement element**

The FcycleStatement element represents a cycle statement.

**Contents model**

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
common attriubutes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
construct_name	text	Specifies the construct name.	O

**6.9 FexitStatement element**

The FexitStatement element represents an exit statement.

**Contents model**

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
common attriubutes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
construct_name	text	Specifies the construct name.	O

**6.10 FreturnStatement element**

The FreturnStatement element represents a return statement.

**Contents model**

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.11 gotoStatement element**

The `gotoStatement` element represents a `go to` statement.

**Contents model**

((params, value)?)

**Child elements**

name	description	R/O
params	Specifies the statement label list of the computed <code>go to</code> statement. Neglected if the <code>label_name</code> attribute is specified.	O
value	Specifies the expression of the computed <code>go to</code> statement. Neglected if the <code>label_name</code> attribute is specified.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
label_name	text	Specifies the statement label.	O

**6.12 statementLabel element**

The `statementLabel` element represents a statement label of the following statement.

**Contents model**

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
label_name	text	Specifies the statement label.	R

**6.13 FselectCaseStatement element**

The FselectCaseStatement element represents a `select case` construct.

**Contents model**

(value, FcaseLabel\*)

**Child elements**

name	description	R/O
value	Specifies the value to select.	R
FcaseLabel	Specifies the <code>case</code> statements included in the element.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
construct_name	text	Specifies the construct name.	O

**6.14 FcaseLabel element**

The FcaseLabel element represents a `case` statement in a `select case` construct. If a FcaseLabel has neither a value element nor an indexRange element, it is assume to be the `case default` statement.

**Contents model**

((value | indexRange)\*, body)

**Child elements**

name	description	R/O
value	Specifies the value of the selector.	O
indexRange	Specifies the range of the selector.	O
body	Specifies the statements included in the element.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
construct_name	text	Specifies the construct name of the corresponding <code>select case</code> statement.	O

### 6.15 FwhereStatement element

The FwhereStatement element represents a `where` statement.

#### Contents model

(condition, then, else?)

#### Child elements

name	description	R/O
condition	Specifies the conditional expression.	R
then	Specifies the statement to execute if the condition falls true.	R
else	Specifies the statement to execute if the condition falls false.	O

#### Attributes

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

### 6.16 FstopStatement element

The FstopStatement element represents a `stop` statement.

#### Contents model

empty

#### Child elements

name	description	R/O
-	-	-

#### Attributes

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
code	text	Specifies a stop code consisting of a string of up to five digits.	O
message	text	Specifies a stop code consisting of a default character constant. Neglected if the code attribute is specified.	O

### 6.17 Input/Output element

#### 6.17.1 FreadStatement, FwriteStatement element

The FreadStatement element and the FwriteStatement element correspond to the `read` statement and the `write` statement respectively.

#### Contents model

(namedValueList, valueList)



**Child elements**

name	description	R/O
<code>namedValueList</code>	Specifies the control list.	R
<code>valueList</code>	Specifies the input or output list.	R

**Attributes**

name	type	description	R/O
<code>common attributes</code>	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**Control list**

The values for the name attribute and the child element of the `namedValue` element are as follows:

name attribute	child element	R/O
<code>unit</code>	Specifies either '*', the scalar default integer expression specifying the external file unit or the scalar default character variable specifying the internal file unit.	R
<code>fmt</code>	Specifies the format specifier, either '*', the scalar default character expression, the statement label or the scalar default integer variable.	R
<code>nml</code>	Specifies the name of the variable group.	O
<code>rec</code>	Specifies the name of the scalar default integer variable.	O
<code>iostat</code>	Specifies the name of the scalar default integer variable.	O
<code>err</code>	Specifies the statement label.	O
<code>end</code>	Specifies the statement label.	O
<code>advance</code>	Specifies the scalar default character expression.	O
<code>size</code>	Specifies the name of the scalar default integer variable.	O
<code>eor</code>	Specifies the statement label.	O
<code>pos</code>	Specifies the scalar integer expression.	O
<code>iomsg</code>	Specifies the scalar default character expression.	O
<code>blank</code>	Specifies the scalar default character expression.	O
<code>pad</code>	Specifies the scalar default character expression.	O
<code>decimal</code>	Specifies the scalar default character expression.	O
<code>delim</code>	Specifies the scalar default character expression.	O
<code>round</code>	Specifies the scalar default character expression.	O
<code>sign</code>	Specifies the scalar default character expression.	O
<code>asynchronous</code>	Specifies the scalar default character expression.	O
<code>id</code>	Specifies the name of the variable.	O

**6.17.2 FprintStatement element**

The `FprintStatement` element corresponds to the `print` statement.

**Contents model**

(`valueList`)

**Child elements**

name	description	R/O
valueList	Specifies the output list.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
format	text	Specifies the format specifier.	R

**6.17.3 FrewindStatement, FendFileStatement, FbackspaceStatement element**

The `FrewindStatement` element, the `FendFileStatement` element and the `FbackspaceStatement` element correspond to the `rewind` statement, the `end file` statement and the `backspace` statement respectively.

**Contents model**

(namedValueList)

**Child elements**

name	description	R/O
namedValueList	Specifies the control list.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**Control list**

The values for the name attribute and the child element of the `namedValue` element are as follows:

name attribute	child element	R/O
unit	Specifies the scalar default integer expression specifying the external file unit.	R
iostat	Specifies the name of the scalar default integer variable.	O
err	Specifies the statement label.	O
iomsg	Specifies the scalar default character expression.	O

**6.17.4 FopenStatement element**

The `FopenStatement` element corresponds to the `open` statement.

**Contents model**

(valueList)

**Child elements**

name	description	R/O
valueList	Specifies the output list.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**Control list**

The values for the name attribute and the child element of the `namedValue` element are as follows:

name attribute	child element	R/O
<code>unit</code>	Specifies the scalar default integer expression specifying the external file unit.	R
<code>iostat</code>	Specifies the name of the scalar default integer variable.	O
<code>err</code>	Specifies the statement label.	O
<code>file</code>	Specifies the name of the scalar default character variable containing the file name.	O
<code>status</code>	Specifies the name of the scalar default character variable.	O
<code>access</code>	Specifies the name of the scalar default character variable.	O
<code>form</code>	Specifies the name of the scalar default character variable.	O
<code>recl</code>	Specifies the name of the scalar default integer variable.	O
<code>blank</code>	Specifies the name of the scalar default character variable.	O
<code>position</code>	Specifies the name of the scalar default character variable.	O
<code>action</code>	Specifies the name of the scalar default character variable.	O
<code>delim</code>	Specifies the name of the scalar default character variable.	O
<code>pad</code>	Specifies the name of the scalar default character variable.	O
<code>newunit</code>	Specifies the name of the scalar default integer variable.	O
<code>iomsg</code>	Specifies the scalar default character expression.	O
<code>decimal</code>	Specifies the scalar default character expression.	O
<code>delim</code>	Specifies the scalar default character expression.	O
<code>encoding</code>	Specifies the scalar default character expression.	O
<code>round</code>	Specifies the scalar default character expression.	O
<code>sign</code>	Specifies the scalar default character expression.	O
<code>asynchronous</code>	Specifies the scalar default character expression.	O

**6.17.5 FcloseStatement element**

The `FcloseStatement` element corresponds to the `close` statement.

**Contents model**

(valueList)

**Child elements**

name	description	R/O
valueList	Specifies the output list.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**Control list**

The values for the name attribute and the child element of the `namedValue` element are as follows:

name attribute	child element	R/O
unit	Specifies the scalar default integer expression specifying the external file unit.	R
iostat	Specifies the name of the scalar default integer variable.	O
err	Specifies the statement label.	O
status	Specifies the name of the scalar default character variable.	O
iomsg	Specifies the scalar default character expression.	O

**6.17.6 FinquireStatement element**

The `FinquireStatement` element corresponds to the `inquire` statement.

**Contents model**

(`namedValueList`, `valueList`)

**Child elements**

name	description	R/O
namedValueList	Specifies the control list.	R
valueList	Specifies the input or output list.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**Control list**

The values for the name attribute and the child element of the `namedValue` element are as follows:

name attribute	child element	R/O
<code>iolength</code>	Specifies the scalar default integer expression. If this is specified, no other specifiers can be specified and the output list must be specified by the <code>valueList</code> element.	O
<code>unit</code>	Specifies the scalar default integer expression specifying the external file unit.	O
<code>file</code>	Specifies the name of the scalar default character variable containing the file name.	O
<code>iostat</code>	Specifies the name of the scalar default integer variable.	O
<code>err</code>	Specifies the statement label.	O
<code>exist</code>	Specifies the name of the scalar default logical variable.	O
<code>opened</code>	Specifies the name of the scalar default logical variable.	O
<code>number</code>	Specifies the name of the scalar default integer variable.	O
<code>named</code>	Specifies the name of the scalar default logical variable.	O
<code>name</code>	Specifies the name of the scalar default character variable.	O
<code>access</code>	Specifies the name of the scalar default character variable.	O
<code>sequential</code>	Specifies the name of the scalar default character variable.	O
<code>direct</code>	Specifies the name of the scalar default character variable.	O
<code>form</code>	Specifies the name of the scalar default character variable.	O
<code>formatted</code>	Specifies the name of the scalar default character variable.	O
<code>unformatted</code>	Specifies the name of the scalar default character variable.	O
<code>recl</code>	Specifies the name of the scalar default integer variable.	O
<code>nextrecl</code>	Specifies the name of the scalar default integer variable.	O
<code>blank</code>	Specifies the name of the scalar default character variable.	O
<code>position</code>	Specifies the name of the scalar default character variable.	O
<code>action</code>	Specifies the name of the scalar default character variable.	O
<code>read</code>	Specifies the name of the scalar default character variable.	O
<code>write</code>	Specifies the name of the scalar default character variable.	O
<code>rewrite</code>	Specifies the name of the scalar default character variable.	O
<code>delim</code>	Specifies the name of the scalar default character variable.	O
<code>pad</code>	Specifies the name of the scalar default character variable.	O
<code>pending</code>	Specifies the name of the scalar default logical variable.	O
<code>pos</code>	Specifies the scalar default integer expression.	O
<code>size</code>	Specifies the scalar default integer expression.	O
<code>iomsg</code>	Specifies the scalar default character expression.	O
<code>decimal</code>	Specifies the scalar default character expression.	O
<code>delim</code>	Specifies the scalar default character expression.	O
<code>encoding</code>	Specifies the scalar default character expression.	O
<code>round</code>	Specifies the scalar default character expression.	O
<code>sign</code>	Specifies the scalar default character expression.	O
<code>stream</code>	Specifies the scalar default character expression.	O
<code>asynchronous</code>	Specifies the scalar default character expression.	O
<code>is</code>	Specifies the scalar integer expression.	O

### 6.17.7 FwaitStatement element

The `Fwait` element corresponds to the `wait` statement.

**Contents model**

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**Control list**

The values for the name attribute and the child element of the `namedValue` element are as follows:

name attribute	child element	R/O
<code>unit</code>	Specifies either '*', the scalar default integer expression specifying the external file unit or the scalar default character variable specifying the internal file unit.	R
<code>err</code>	Specifies the statement label.	O
<code>end</code>	Specifies the statement label.	O
<code>eor</code>	Specifies the statement label.	O
<code>iostat</code>	Specifies the scalar default integer variable.	O
<code>iomsg</code>	Specifies the scalar default character expression.	O
<code>id</code>	Specifies the scalar default integer expression.	O

**6.17.8 FflushStatement element**

The `FflushStatement` element corresponds to the `flush` statement.

**Contents model**

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**Control list**

The values for the name attribute and the child element of the `namedValue` element are as follows:

name attribute	child element	R/O
<code>unit</code>	Specifies the scalar default integer expression specifying the external file unit.	R
<code>err</code>	Specifies the statement label.	O
<code>iomsg</code>	Specifies the scalar default character expression.	O
<code>iostat</code>	Specifies the scalar default integer variable.	O

**6.18 FformatDecl element**

The `FformatDecl` element represents a `format` statement.

**Contents model**

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
<code>common attributes</code>	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
<code>format</code>	text	Specifies the character string representing the format specification.	R

**6.19 FdataDecl element**

The `FdataDecl` element represents a `data` statement.

**Contents model**

((`varList`, `valueList`)+)

**Child elements**

name	description	R/O
<code>varList</code>	Specifies the object list.	R
<code>valueList</code>	Specifies the value list.	R

**Attributes**

name	type	description	R/O
<code>common attributes</code>	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.20 FnamelistDecl element**

The FnamelistDecl element represents a `namelist` statement.

**Contents model**

(varList+)

**Child elements**

name	description	R/O
varList	Specifies the name of the variable group and the list of the element variables.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.21 FequivalenceDecl element**

The FequivalenceDecl element represents an `equivalent` statement.

**Contents model**

((varRef, varList)+)

**Child elements**

name	description	R/O
varRef	Specifies the object. The object must be a variable, array element or substring.	R
varList	Specifies the object list.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.22 FcommonDecl element**

The FcommonDecl element represents a `common` statement.

**Contents model**

(varList+)

**Child elements**

name	description	R/O
varList	Specifies the name of the common block and the list of the variable names.	R



**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.23 FentryDecl element**

The FentryDecl element represents an **entry** statement.

**Contents model**

(name, symbols?, params?)

**Child elements**

name	description	R/O
name	Specifies the name of entry.	R
symbols	Specifies the symbols included in the element.	O
params	Specifies the dummy arguments.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.24 Allocation element**

The following elements define the allocate statements.

**6.24.1 FallocateStatement element**

The FallocateStatement element represents an **allocate** statement.

**Contents model**

(alloc+, alloc\_opt\*)

**Child elements**

name	description	R/O
alloc	Specifies the allocation list.	R
alloc_opt	Specifies the allocation option.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
stat_name	text	Specifies the name of status variable.	O
type	text	Specifies the type identifier if the type is specified.	O

**Example**

The `FallocateStatement` element for the `allocate` statement below is as follows:

```

Class(t),pointer :: x(:)
Class(*),pointer :: a(:),b(:)
allocate(et::x(100),a(100))
allocate(b,source=a,stat=s)

```

```

XcodeML/Fortran
<FallocateStatement type="ID_OF_ET" >
  <alloc>
    <Var>x</Var>
    <arrayIndex>
      <FintConstant>100</FintConstant>
    </arrayIndex>
  </alloc>
  <alloc>
    <Var>a</Var>
    <arrayIndex>
      <FintConstant>100</FintConstant>
    </arrayIndex>
  </alloc>
</FallocateStatement>
<FallocateStatement>
  <alloc>
    <Var>b</Var>
  </alloc>
  <alloc\_opt kind="source">
    <Var>a</Var>
  </alloc\_opt>
  <alloc\_opt kind="stat">
    <Var>s</Var>
  </alloc\_opt>
</FallocateStatement>

```

**6.24.2 FdeallocateStatement element**

The `FdeallocateStatement` element represents a `deallocate` statement.

**Contents model**

(`alloc+`, `alloc_opt*`)

**Child elements**

name	description	R/O
<code>alloc</code>	Specifies the name of the allocate object list.	R
<code>alloc_opt</code>	Specifies the allocation option.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
stat_name	text	Specifies the name of status variable.	O

**6.24.3 FnullifyStatement element**

The FnullifyStatement element represents a nullify statement.

**Contents model**

(alloc+)

**Child elements**

name	description	R/O
alloc	Specifies the pointer object list.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.25 FpragmaStatement element**

The FpragmaStatement element represents a directive of OpenMP 3.0 beginning with ' !\$ ' or ' !\$OMP ' . It has the content as text data.

**Contents model**

(#PCDATA)

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.26 FcontainsStatement element**

The FcontainsStatement element represents a contains statement.

**Contents model**

(FfunctionDefinition+)

**Child elements**

name	description	R/O
FfunctionDefinition	Specifies the functions and subroutines included in the element.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.27 condition element**

The condition element represents a conditional expression.

**Contents model**

(exprModel)

**Child elements**

name	description	R/O
exprModel	Specifies the expression. Refer to "9.4 exprModel".	R

**Attributes**

name	type	description	R/O
-	-	-	-

**6.28 then element**

The then element represents a block executed if the condition falls true.

**Contents model**

(body)

**Child elements**

name	description	R/O
body	Specifies the statements included in the then statement.	R

**Attributes**

name	type	description	R/O
-	-	-	-

**6.29 else element**

The else element represents a block executed if the condition falls false.

**Contents model**

(body)

**Child elements**

name	description	R/O
<b>name</b>	Specifies the statements included int the <b>else</b> statement.	R

**Attributes**

name	type	description	R/O
-	-	-	-

**6.30 alloc element**

The **alloc** element represents an object in the allocation list of the **allocate** and **deallocate** statement and in the pointer list of the **nullify** statement.

**Contents model**

((FmemberRef | Var), (arrayIndex | indexRange)?, coShape?\*)

**Child elements**

name	description	R/O
<b>FmemberRef</b>	Specifies the reference to the component of the structure.	R
<b>Var</b>	Specifies the variable.	R
<b>arrayIndex</b>	Specified if the type element is the array type and the size is expressed by the number of the elements. Can be repeated the number of the dimension times together with the <b>indexRange</b> element. Neglected if the element is a child element of the <b>FnullifyStatement</b> .	O
<b>indexRange</b>	Specified if the type element is the array type and the size is expressed by the upper and lower bounds. Can be repeated the number of the dimension times together with the <b>arrayIndex</b> element. Neglected if the element is a child element of the <b>FnullifyStatement</b> .	O
<b>coShape</b>	Specified if the element represents the allocation of a coarray.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**6.31 allocOpt element**

The **allocOpt** element represents an **alloc** option.

**Contents model**

(exprModel+)

**Child elements**

name	description	R/O
<b>exprModel</b>	Specifies the expression for the value of the element.	R

**Attributes**

name	type	description	R/O
kind	text	Specifies "errmsg", "mold", "source" or "stat" as the kind of the alloc option.	R

**6.32 forallStatement element**

The forallStatement element represents a forall construct.

**Contents model**

((var, indexRange)+, condition?, body)

**Child elements**

name	description	R/O
var	Specifies the index variable.	O
indexRange	Specifies the value range of the index variable.	O
condition	Specifies the masking condition.	O
body	Specifies the content statements of the forall construct.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
type	text	Specifies the type identifier if the type is specified.	O
construct_name	text	Specifies the construct name.	O

**Example**

The forallStatement element for the forall construct below is as follows:

Fortran 2008

```
forall (i=1:n,j=1:m,ix(i,j)>0)
  ix(i,j)=0
end forall
```

XcodeML/Fortran

```
<forallStatement>
  <Var>i</Var>
  <indexRange>
    <lowerBound>
      <FintConstant>1</FintConstant>
    </lowerBound>
    <upperBound>
      <Var>n</Var>
    </upperBound>
    <step>
      <FintConstant>1</FintConstant>
    </step>
  </indexRange>
```

```

15     <Var>j </Var>
    <indexRange>
      <lowerBound>
        <FintConstant>1</FintConstant>
      </lowerBound>
      <upperBound>
20         <Var>m</Var>
      </upperBound>
      <step>
        <FintConstant>1</FintConstant>
      </step>
25 </indexRange>
    <condition>
      <logGTEExpr>
        <FarrayRef>
          <varRef>
30             <Var>ix</Var>
          </varRef>
          <arrayIndex>
            <Var>i</Var>
          </arrayIndex>
35          <arrayIndex>
            <Var>j</Var>
          </arrayIndex>
        </FarrayRef>
        <FintConstant>0</FintConstant>
40      </logGTEExpr>
    </condition>
    <body>
      <FassignStatement">
        <FarrayRef>
45          <varRef>
            <Var>ix</Var>
          </varRef>
          <arrayIndex>
            <Var>i</Var>
50          </arrayIndex>
          <arrayIndex>
            <Var>j</Var>
          </arrayIndex>
        </FarrayRef>
55        <FintConstant>0</FintConstant>
      </FassignStatement>
    </body>
  </FdoStatement>

```

### 6.33 FdoConcurrentStatement element

The `FdoConcurrentStatement` element represents a `do concurrent` statement.

**Contents model**

((var, indexRange)+, condition?, body?)

**Child elements**

name	description	R/O
var	Specifies the do variable.	R
indexRange	Specifies the value range of the do variable.	R
condition	Specifies the masking condition.	O
body	Specifies the content statements.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
type	text	Specifies the type identifier if the type is specified.	O
construct_name	text	Specifies the construct name.	O

**6.34 selectTypeStatement element**

The selectTypeStatement element represents a select type construct.

**Contents model**

(id, typeGuard\*)

**Child elements**

name	description	R/O
id	Specifies the variable or the expression (and its associate name) to test the type.	R
typeGuard	Specifies the type guard statement in the construct.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
construct_name	text	Specifies the construct name.	O

**Example**

The selectTypeStatement element for the select type construct below is as follows:

Fortran 2008

```

class(t) x
...
select type(p=>x)
type is(t1)
...
type is(t2)
...

```



```

class is(t3)
  ...
10 type default
  ...
end select

```

XcodeML/Fortran

```

<selectTypeStatement>
  <id>
    <name>p</name>
    <value>
      <Var>x</Var>
    </value>
  </id>
  <typeGuard kind="TYPE_IS" type="TYPE_ID_OF_T1">
    <body>
      ...
    </body>
  </typeGuard>
  <typeGuard kind="TYPE_IS" type="TYPE_ID_OF_T2">
    <body>
      ...
    </body>
  </typeGuard>
  <typeGuard kind="CLASS_IS" type="TYPE_ID_OF_T3">
    <body>
      ...
    </body>
  </typeGuard>
  <typeGuard kind="TYPE_DEFAULT">
    <body>
      ...
    </body>
  </typeGuard>
</selectTypeStatement>

```

### 6.35 typeGuard element

The typeGuard element represents a type guard statement in the select type construct.

#### Contents model

(body)

#### Child elements

name	description	R/O
body	Specifies the content statements.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
kind	text	Specifies "TYPE_IS", "CLASS_IS" or "TYPE_DEFAULT" as the kind of the type guard.	R
type	text	Specifies the type identifier to test.	O

**6.36 syncAllStatement element**

The `syncAllStatement` element represents a `sync all` statement.

**Contents model**

(`syncStat*`)

**Child elements**

name	description	R/O
<code>syncStat</code>	Specifies the stat specifiers.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.37 syncImagesStatement element**

The `syncImagesStatement` element represents a `sync image` statement.

**Contents model**

(`expModel?`, `syncStat*`)

**Child elements**

name	description	R/O
<code>exprModel</code>	Specifies the expression representing the image. Omitted if '*' is specified.	O
<code>syncStat</code>	Specifies the stat specifiers.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.38 syncMemoryStatement element**

The `syncMemoryStatement` element represents a `sync memory` statement.

**Contents model**

(syncStat\*)

**Child elements**

name	description	R/O
syncStat	Specifies the stat specifiers.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.39 lock/unlockStatement element**

The lock/unlockStatement element represents a lock/unlock statement.

**Contents model**

(Var?, syncStat\*)

**Child elements**

name	description	R/O
Var	Specifies the lock variable.	O
syncStat	Specifies the stat specifiers.	O

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-

**6.40 syncStat element**

The syncStat element represents a stat specifier.

**Contents model**

(Var)

**Child elements**

name	description	R/O
Var	Specifies the variable.	O

**Attributes**

name	type	description	R/O
kind	text	Specifies "STAT", "ERRMSG" or "ACQUIRED_LOCK" as the kind of the stat specifier.	R

**6.41 criticalStatement element**

The `criticalStatement` element represents a `critical` construct.

**Contents model**

(body)

**Child elements**

name	description	R/O
body	Specifies the content statements.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
construct_name	text	Specifies the construct name.	O

**6.42 associateStatement element**

The `associateStatement` element represents a `associate` construct.

**Contents model**

(symbols, body)

**Child elements**

name	description	R/O
symbols	Specifies the bindings within the construct.	R
body	Specifies the content statements.	R

**Attributes**

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
construct_name	text	Specifies the construct name.	O

**Example**

The `associateStatement` element for the `associate` statement below is as follows:

Fortran 2008
<pre> associate(x=&gt;... , y=&gt;...)   ... = x   ... = y end associate </pre>

		XcodeML/Fortran
5		<pre> &lt;associateStatement&gt;   &lt;symbols&gt;     &lt;id&gt;       &lt;name&gt;x&lt;/name&gt;       &lt;value&gt;         ...       &lt;/value&gt;     &lt;/id&gt;     &lt;id&gt;       &lt;name&gt;y&lt;/name&gt;       &lt;value&gt;         ...       &lt;/value&gt;     &lt;/id&gt;   &lt;/symbols&gt;   &lt;body&gt;     ...   &lt;/body&gt; &lt;/associateStatement&gt; </pre>
10		
15		

### 6.43 blockStatement element

The blockStatement element represents a block construct.

#### Contents model

(symbols?, declarations?, body)

#### Child elements

name	description	R/O
symbols	Specifies the symbols included in the element.	O
declarations	Specifies the declarations included in the block construct.	O
body	Specifies the content statements.	R

#### Attributes

name	type	description	R/O
common attributes	-	Refer to "9.2 Common attributes of the definition / declaration / statement element".	-
construct_name	text	Specifies the construct name.	O

#### Example

The blockStatement element for the block construct below is as follows:

		Fortran 2008
		<pre> block   integer i   real x </pre>

```

...
5 end block

```

XcodeML/Fortran

```

<blockStatement>
  <symbols>
    <id type="Fint" sclass="flocal">
      <name>i</name>
5    </id>
    <id type="Freal" sclass="flocal">
      <name>a</name>
    </id>
  </symbols>
10 <declarations>
    <varDecl>
      <name type="Fint">i</name>
    </varDecl>
    <varDecl>
15    <name type="Freal">a</name>
    </varDecl>
  </declarations>
  <body>
    ...
20 </body>
</blockStatement>

```

## 7 Expression element

This chapter describes the XML elements corresponding to the syntax elements for expressions in Fortran. Each element has a `type` element attached, which can be used to extract the data type information of the expression.

### 7.1 Constant element

The elements representing constants are described below.

#### 7.1.1 `FintConstant`, `FrealConstant`, `FcharacterConstant`, `FlogicalConstant` element

The `FintConstant`, `FrealConstant`, `FcharacterConstant` and `FlogicalConstant` element represents a constant of the integer, real, character and logical type respectively.

The contents of the elements are shown below.

element	format of the content
<b>FintConstant</b>	Specifies a constant that has an integer value; a decimal or hexadecimal (starting with "0x") number.
<b>FrealConstant</b>	32bit hexadecimal (starting with 0x) two numbers representing an IEEE754 floating-point number.
<b>FcharacterConstant</b>	A character string.
<b>FlogicalConstant</b>	1(or <b>true</b> ) representing true of the logical value, or 0(or <b>false</b> ) representing false.

### Contents model

(#PCDATA)

### Child elements

name	description	R/O
-	-	-

### Attributes

name	type	description	R/O
<b>type</b>	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O
<b>kind</b>	text	Specifies the kind parameter of the type element.	O

#### 7.1.2 FcomplexContant element

The **FcomplexContant** element represents a constant of the **complex** type.

### Contents model

(**FrealConstant**, **FrealConstant**)

### Child elements

name	description	R/O
<b>FrealConstant</b>	Specifies a couple of a constant of the <b>Freal</b> type.	R

### Attributes

name	type	description	R/O
<b>type</b>	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O

## 7.2 Array constructor element

### 7.2.1 FarrayConstructor element

The **FarrayConstructor** element represents an array constructor.

### Contents model

(**exprModel**)\*

**Child elements**

name	description	R/O
exprModel	Specifies the expression of the value of the array element. Refer to "9.4 exprModel".	O

**Attributes**

name	type	description	R/O
type	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O

**7.3 Structure constructor element****7.3.1 Constant element**

The `FstructConstructor` element represents a structure constructor.

**Contents model**

(typeParamValues?, (exprModel)\*)

**Child elements**

name	description	R/O
typeParamValues	Specifies the value of the type parameter.	O
exprModel	Specifies the expressions representing the values of the structure components. Refer to "9.4 exprModel".	R

**Attributes**

name	type	description	R/O
type	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O

**Example**

The `FstructConstructor` element for the type `S` below is as follows:

```

----- Fortran 2008 -----
type S(alen,blen,clen)
real,length:: alen,blen,clen
real::a(alen) = 0.0
real::b(blen) = 0.0
5 real::b(clen) = 0.0
end type
... = S(100,100,100) (b=1.0,c=2.0)

```

```

----- XcodeML/Fortran -----
<FstructConstructor type="TYPE_ID_OF_S">
<typeParamValues>
  <FintConstant type="Fint">100</FintConstant>
  <FintConstant type="Fint">100</FintConstant>
5 <FintConstant type="Fint">100</FintConstant>

```



```

</typeParamValues>
<namedValue name="b">
  <FrealConstant>1.0</FrealConstant>
</namedValue>
10 <namedValue name="c">
  <FrealConstant>2.0</FrealConstant>
</namedValue>
</FstructConstructor>

```

## 7.4 Elements referencing variables

The elements representing a reference to a variable are described below.

### 7.4.1 Var element

The `Var` element represents a name of the variable. Specifies the name of the variable as the content.

#### Contents model

(#PCDATA)

#### Child elements

name	description	R/O
-	-	-

#### Attributes

name	type	description	R/O
<code>type</code>	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O
<code>scope</code>	text	Specifies one of "local" for a local variable, "global" for a global variable, "param" for a dummy argument. Note: "global" is not used in XcodeML/Fortran, however.	R

### 7.4.2 FmemberRef element

The `FmemberRef` element represents a reference to a structure component.

#### Contents model

(varRef)

#### Child elements

name	description	R/O
<code>varRef</code>	Specifies the reference to the structure.	R

**Attributes**

name	type	description	R/O
<code>type</code>	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O
<code>member</code>	text	Specifies the name of the component.	R
<code>is_complex_part</code>	bool	<code>true</code> if referring to the <code>complex</code> type partially. In this case, the member name(the value of the <code>member</code> attribute) is either "RE" or "IM".	O

**7.4.3 FcoArrayRef element**

The `FcoArrayRef` element represents a reference to a coarray.

**Contents model**

(`varRef`, (`arrayIndex`)<sup>+</sup>)

**Child elements**

name	description	R/O
<code>varRef</code>	Specifies the coindexed named object.	R
<code>arrayIndex</code>	Specifies the image selector.	R

**Attributes**

name	type	description	R/O
<code>type</code>	text	Specifies the type name identifier corresponding to the referenced coarray in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O

**7.4.4 FarrayRef element**

The `FarrayRef` element represents a reference to an array section or an array element. In the case representing a reference to an array element, specifies all of the index elements by the `arrayIndex` element.

**Contents model**

(`varRef`, (`arrayIndex` | `indexRange` | `FarrayConstructor` | `FarrayRef`)<sup>\*</sup>)

**Child elements**

name	description	R/O
<code>varRef</code>	Specifies the reference to the array.	R
<code>arrayIndex</code>	Specified if the element list of the array section is expressed by the subscripts. Can be repeated the number of the dimension times together with other elements.	O
<code>indexRange</code>	Specified if the element list of the array section is expressed by the upper and lower bounds. Can be repeated the number of the dimension times together with other elements.	O
<code>FarrayConstructor</code>	Specified if the element list of the array section is expressed by the vector subscript by the array constructor. Can be repeated the number of the dimension times together with other elements.	O
<code>FarrayRef</code>	Specified if the element list of the array section is expressed by the vector subscript by the array section. Can be repeated the number of the dimension times together with other elements.	O

**Attributes**

name	type	description	R/O
<code>type</code>	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O

**7.4.5 FcharacterRef element**

The `FcharacterRef` element represents a reference to a character substring.

**Contents model**

(`varRef`, `indexRange`?)

**Child elements**

name	description	R/O
<code>varRef</code>	Specifies the reference to the character variable.	R
<code>indexRange</code>	Specifies the element list of the character string by the upper and lower bounds.	O

**Attributes**

name	type	description	R/O
<code>type</code>	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O

**7.4.6 varRef element**

The `varRef` element represents a reference to a variable.

**Contents model**

(`Var` | `FmemberRef` | `FarrayRef` | `FcharacterRef` | `FcoArrayRef`)

**Child elements**

name	description	R/O
<b>Var</b>	Specifid if the element references the variable.	R
<b>FmemberRef</b>	Specifid if the element references the structure component.	R
<b>FarrayRef</b>	Specifid if the element references the array section of the array.	R
<b>FcharacterRef</b>	Specifid if the element references the character substring of the character string.	R
<b>FcoArrayRef</b>	Specifid if the element references the coarray.	R

**Attributes**

name	type	description	R/O
<b>type</b>	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O

**7.5 Function call****7.5.1 functionCall element**

The `functionCall` element represents a function / subroutine call.

**Contents model**

((`name|FmemberRef`), `arguments?`)

**Child elements**

name	description	R/O
<b>name</b>	Specifies the name of the function or subroutine.	R
<b>FmemberRef</b>	Specifies the name of the function or subroutine (in the case of the component of the structure or the type bound procedure).	R
<b>arguments</b>	Specifies the expressions of the arguments.	O

**Attributes**

name	type	description	R/O
<b>type</b>	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O
<b>is_intrinsic</b>	bool	Specifies if the intrinsic function / subroutine call.	O

**7.5.2 arguments element****Contents model**

(`exprModel`)\*

**Child elements**

name	description	R/O
<b>exprModel</b>	Specifies the expressions of the arguments. Refer to "9.4 <code>exprModel</code> ".	O

**Attributes**

name	type	description	R/O
-	-	-	-

**7.6 Binary operation element**

The elements representing the binary operators are described below.

element	operator	operation
plusExpr	+	addition
minusExpr	-	subtraction
mulExpr	*	multiplication
divExpr	/	division
FpowerExpr	**	exponentiation
FconcatExpr	//	concatenation
logEQExpr	== .EQ.	equal
logNEQExpr	/= .NE.	not equal
logGEEExpr	>= .GE.	greater than or equal
logGTExpr	> .GT.	greater than
logLEExpr	<= .LE.	less than or equal
logLTExpr	< .LT.	less than
logAndExpr	.AND.	logical intersection
logOrExpr	.OR.	logical union
logEQVExpr	.EQV.	logical equivalence
logNEQVExpr	.NEQV.	logical non-equivalence
userBinaryExpr	arbitrary	depends on the interface.

**Contents model**

(exprModel, exprModel)

**Child elements**

name	description	R/O
exprModel	Specifies the left-hand expression as the first operand, the right-hand expression as the second operand. Refer to "9.4 exprModel".	R

**Attributes**

name	type	description	R/O
type	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O

## 7.7 Unary operation element

The elements representing the unary operators are described below. No element is prepared for the unary operator '+' because it is omitted by the canonicalization.

element	operator	operation
unaryMinusExpr	-	sign inversion
logNotExpr	.NOT.	logical complement
userUnaryExpr	arbitrary	depends on the interface.

### Contents model

(exprModel)

### Child elements

name	description	R/O
exprModel	Specifies the operand expression. Refer to "9.4 exprModel.	R

### Attributes

name	type	description	R/O
type	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	O

## 8 General element

### 8.1 kind element

The kind element represents a kind parameter of the type.

### Contents model

(#PCDATA)

### Child elements

name	description	R/O
-	-	-

### Attributes

name	type	description	R/O
-	-	-	-

### 8.2 id element

The id element represents an identifier of the name of the variable, array, structure component, dummy argument of the function or subroutine, etc.

### Contents model

(name)

**Child elements**

name	description	R/O
name	Specifies the name of the identifier(the name of the variable if the identifier corresponds to a variable etc.).	R

**Attributes**

name	type	description	R/O
sclass	text	Specifies the storage class, one of 'auto', 'param', 'extern', 'extern_def', 'label', 'tagname'.	R
type	text	Specifies the type name identifier in XcodeML/Fortran. Refer to "9.1 The data type name identifier".	R

**Example**

The symbol table entry of the variable "xyz" in "integer xyz" is as follows:

```

XcodeML/Fortran
<id sclass="extern_def" type="Fint">
  <name>xyz</name>
</id>
```

The symbol table entry of the function "foo" in "function foo" is as follows: Here, "F06f168" is the type\_id of the data type of "foo".

```

XcodeML/Fortran
<id sclass="extern_def" type="F06f168">
  <name>foo</name>
</id>
```

**8.3 name element**

The name element specifies the name of a variable or a type name etc.

**Contents model**

(#PCDATA)

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
type	text	Specifies the type name identifier in XcodeML/Fortran.	R

**8.4 value element**

The value element represents an arbitrary value expressed by an expression.

**Contents model**

(exprModel)

**Child elements**

name	description	R/O
exprModel	Specifies the expression of the value. Refer to "9.4 exprModel".	R

**Attributes**

name	type	description	R/O
repeat_counttyp	text	Specifies the repeat count of the element. Valid only in the initialization expression of the <b>data</b> statement.	O

**8.5 kind params**

The **param** element represents a dummy argument list of the function or subroutine. The element is also used to represent a statement label list in the **go to** statement.

**Contents model**

(name\*)

**Child elements**

name	description	R/O
name	Specifies the names of the dummy arguments or the statement labels. In the case of the statement labels, the type attribute of the name element must be 'Fint'.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**8.6 len element**

The **len** element represents an arbitrary length expressed by an expression.

**Contents model**

(exprModel)

**Child elements**

name	description	R/O
exprModel	Specifies the expression of the length. Refer to "9.4 exprModel".	R



**Attributes**

name	type	description	R/O
<code>is_assumed_shape</code>	bool	true if no length is specified("(:)").	O
<code>is_assume_size</code>	bool	true if the length is the assumed-size.	O

**8.7 body element**

The body element represents a block of statements.

**Contents model**

(statementModel\*)

**Child elements**

name	description	R/O
<code>statementModel</code>	Specifies arbitrary statements.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**8.8 rename element**

The `rename` element represents a local name and a name accessed in the `use` statement.

**Contents model**

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
<code>use_name</code>	text	Specifies the name accessed.	R
<code>local_name</code>	text	Specifies the local name.	R
<code>is_operator</code>	bool	true if the element represents the rename of the defined operator.	O

**8.9 renamable element**

The `renamable` element represents a local name and a name accessed, which are omissible, in the `use` statement with the only option.

**Contents model**

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
use_name	text	Specifies the name accessed.	R
local_name	text	Specifies the local name.	O
is_operator	bool	true if the element represents the rename of the defined operator.	O

**8.10 arrayIndex element**

The `arrayIndex` element represents an arbitrary index value(or size) expressed by an expression.

**Contents model**

(`exprModel`)

**Child elements**

name	description	R/O
<code>exprModel</code>	Specifies the expression of the value. Refer to "9.4 <code>exprModel</code> ".	R

**Attributes**

name	type	description	R/O
-	-	-	-

**8.11 indexRange element**

The `indexRange` element represents an arbitrary range of the subscript(or size) expressed by an upper and lower bound and a step. The meaning of the element depends on the kind of the parent element.

**Contents model**

(`lowerBound?`, `upperBound?`, `step?`)

**Child elements**

name	description	R/O
<code>lowerBound</code>	Specifies the lower bound. The kind of the parent element determines the default value.	O
<code>upperBound</code>	Specifies the upper bound. The kind of the parent element determines the default value.	O
<code>step</code>	Specifies the step. May be neglected depending on the kind of the parent element.	O

**Attributes**

name	type	description	R/O
is_assumed_shape	bool	true in the case of the assumed-shape array.	O
is_assumed_size	bool	true in the case of the assumed-size array.	O

**8.12 lowerBound element**

The lowerBound element represents a lower bound of a range.

**Contents model**

(exprModel)

**Child elements**

name	description	R/O
exprModel	Specifies the expression of the lower bound. Refer to "9.4 exprModel".	R

**Attributes**

name	type	description	R/O
-	-	-	-

**8.13 upperBound element**

The upperBound element represents an upper bound of the range.

**Contents model**

(exprModel)

**Child elements**

name	description	R/O
exprModel	Specifies the expression of the upper bound. Refer to "9.4 exprModel".	R

**Attributes**

name	type	description	R/O
-	-	-	-

**8.14 step element**

The step element represents a stride of a range.

**Contents model**

(exprModel)

**Child elements**

name	description	R/O
exprModel	Specifies the expression of the stride. Refer to "9.4 exprModel".	R

**Attributes**

name	type	description	R/O
-	-	-	-

**8.15 FdoLoop element**

The `FdoLoop` element represents an implied-do. The element defines all the implied-does in the data statement, the I/O statement and the array constructor.

**Contents model**

(Var, indexRange, value+)

**Child elements**

name	description	R/O
Var	Specifies the do variable.	R
indexRange	Specifies the upper and lower bounds.	R
value	Specifies the expression element for the value of the implied-do.	R

**Attributes**

name	type	description	R/O
-	-	-	-

**8.16 namedValue element**

The `namedValue` element represents a value with a keyword.

**Contents model**

empty

**Child elements**

name	description	R/O
-	-	-

**Attributes**

name	type	description	R/O
name	text	Specifies the keyword.	R
value	text	Specifies the value.	R

**8.17 namedValueList element**

The namedValueList element represents a value list with keywords.

**Contents model**

(namedValue\*)

**Child elements**

name	description	R/O
namedValue	Specifies the value with the keyword.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**8.18 valueList element**

The valueList element represents a value list expressed by an expression.

**Contents model**

(value\*)

**Child elements**

name	description	R/O
value	Specifies the expression for the value.	O

**Attributes**

name	type	description	R/O
-	-	-	-

**8.19 varList element**

The varList element represents a list.

**Contents model**

(varRef | FdoLoop)\*

**Child elements**

name	description	R/O
varRef	Specifies the item expressed by the reference to the variable.	O
FdoLoop	Specifies the list of the items expressed by the implied-do.	O

**Attributes**

name	type	description	R/O
name	text	Specifies the namelist group name if the parent is the FnamelistDecl element.	O

## 9 Common element

The definitions commonly used in an arbitrary element are shown below.

### 9.1 The data type name identifier

In the program, the data type is identified by the data name. The name is one of the basic data type names or the derived data type names below.

type name	description
<b>Fint</b>	integer type
<b>Freal</b>	real type
<b>Fcomplex</b>	complex type
<b>Flogical</b>	logical type
<b>Fcharacter</b>	character type
<b>Fvoid</b>	void type; represents the type of the return value of the subroutine.
<b>others</b>	derived type; expressed by an arbitrary string consisting of alphabets and numbers different from the names of the basic data types. It must be unique in the program.

### 9.2 Common attributes of the definition / declaration / statement element

Every definition, declaration and statement elements may have the following attributes.

#### Attributes

name	type	description	R/O
<b>lineno</b>	text	Specifies the line number in the Fortran program.	R
<b>file</b>	text	Specifies the source code name of the Fortran program.	R

### 9.3 lValueModel model

The lValueModel model is commonly used by the elements which refer to the variable as the lvalue.

#### Contents model

(Var | FarrayRef | FcharacterRef | FmemberRef | FcoArrayRef)

#### Element

Refer to the specification of each element.

### 9.4 exprModel model

The exprModel model is commonly used by the elements which refer to the expression.

#### Contents model

(FintConstant | FrealConstant | FcomplexConstant | FcharacterConstant | FlogicalConstant | FarrayConstructor | FstructConstructor | Var | FarrayRef | FcharacterRef | FmemberRef | FcoArrayRef | varRef | functionCall | plusExpr | minusExpr | mulExpr | divExpr | FpowerExpr | FconcatExpr | logEQExpr | logNEQExpr | logGEEExpr | logGTExpr | logLEExpr

| logLTExpr | logAndExpr | logOrExpr | logEQVExpr | logNEQVExpr | unaryMinusExpr  
 | logNotExpr | userBinaryExpr | userUnaryExpr | FdoLoop)

### Element

Refer to the specification of each element.

## 9.5 statementModel model

The statementModel model is commonly used by the elements which refer to the statement.

### Contents model

(FassignStatement | exprStatement | FpointerAssignStatement | FifStatement | FdoStatement  
 | FdoWhileStatement | continueStatement | FcycleStatement | FexitStatement | FreturnStatement  
 | gotoStatement | statementLabel | FselectCaseStatement | FcaseLabel | FwhereStatement  
 | FstopStatement | FreadStatement | FwriteStatement | FprintStatement | FrewindStatement  
 | FendFileStatement | FbackspaceStatement | FopenStatement | FcloseStatement | FinquireStat  
 | FformatDecl | FdataDecl | FentryDecl | FallocateStatement | FdeallocateStatement  
 | FnullifyStatement | FpragmaStatement | FcontainsStatement)

### Element

Refer to the specification of each element.

## 10 Miscellaneous

### 10.1 Intrinsic procesures

Intrinsic procesures are treated as external functions or subroutines without `extern` declaration.

### 10.2 Treatment of comentms and pragmas

There is no element which represents comments. However, the pragma elements for the directives of OpenMP 3.0 are kept by the `FpragmaStatement` elements.

### 10.3 Canonicalization

In this specification of XcodeML/Fortran, codes are assumed to be altered in the actual conversion process by the frontend within the range where the meanings of the codes are not changed. The alterations are made as follows:

- As for the `implicit` statement, declares all the variables explicitly and adds the `implicit none` statement.
- The `parameter` statement, the `dimension` statement, the `allocatable` statement, the `pointer` statement, the `target` statement, the `intent` statement and the `save` statement saving a variable are unified into a type declaration of the variable.
- The `parameter` variable initialized by a constant is replaced by the `constatnt`.
- The `statement function` is inlined.
- The unary operator '+' is omitted.

- Any comments other than the directives of OpenMP 3.0 are removed.

Accordingly, some Fortran 90 syntax elements such as the `statement function` have no corresponding elements in XcodeML/Fortran.

## 10.4 Restrictions

This specification of XcodeML/Fortran does not support the following Fortran 90 syntaxes because they do not appear in both of NBP and SPEC benchmarks.

- The `block data` statement, the `end block data` statement.
- The `optional` statement within the `module` declaration.
- The binary, octal and hexadecimal integer expression by the `BOZ` literal.

On the other hand, the following 6 items from the obsolescent features in Fortran 90 are not supported.

- The `do` variable and the expressions of type default real or double precision real, and the shared `do-loop` termination.
- The `assign` statement, the `assigned go to` statement and the `assigned format`.
- Branching to an `end if` statement.
- The alternate `return` and the `alternate return` statement.
- The `pause` statement.
- The `H edit` descriptor.

## 11 Bibliography

1. The XcodeML specification(version 0.8J).
2. The report on the usage of Fortran 90/95 features in NBP/SPEC.
3. The report on the usage of Fortran 90/95 intrinsic procedures in NBP/SPEC.
4. The International Standard : ISO/IEC 1539-1991 The Programming Languages FORTRAN
5. The Japanese Industrial Standard : JIS X 3001-1994 Programing Language FORTRAN