ASIS 95

The ASIS interface for Ada 95

ASIS 95 Panel, Ada-Europe ´96

Jesper Jørgensen, DDC-I 16-06-96 1

This is just the title page.

ASIS Interface

A set of packages used for querying the Ada environment

» Environment
  – compilation units as black boxes
  – relations between compilation units

» Compilation units as white boxes
  – syntactic information
  – semantic information

Environment:

Compilation units can be fetched by name or by particular relation to some other already fetched compilation unit.

Black boxes: get attributes, unit kind, text name

Relations: body-of, child-of, semantic dependences, ...

Compilation units:

Syntactic:

Fetch unit declaration from compilation unit,

Traverse nodes of syntax tree up/down/left/right,

Classification of tree nodes

Semantic:

Find declaring occurrence from a name,

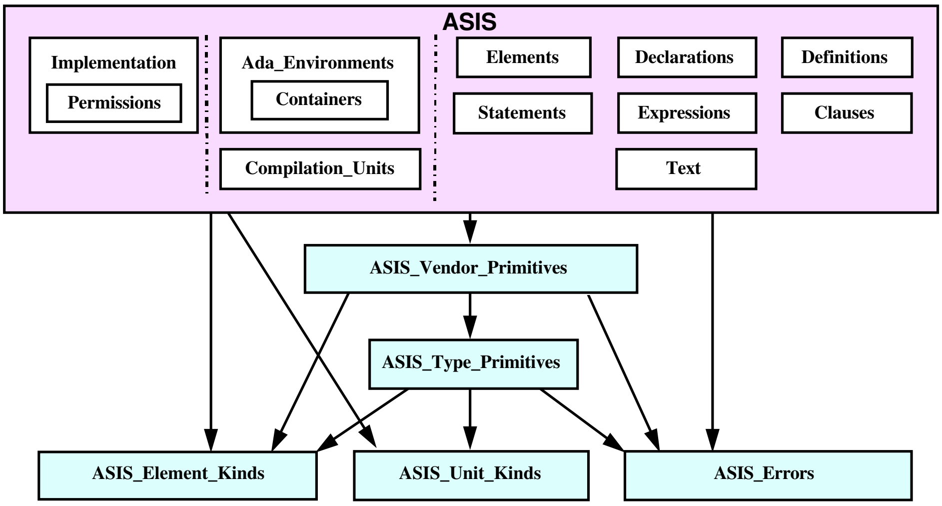
Find corresponding declaration:

spec - body

private/incomplete type - full type

deferred constant - full constant

ASIS Overview



Jesper Jørgensen, DDC-I 16-06-96 3

This is a slightly simplified picture of the ASIS package hierarchy.

Child packages are shown nested, while individual packages are shown as “free-floating” boxes.

Semantic dependencies are shown as arrows.

An ASIS application *only* interfaces ASIS + children which “repeat” all types of lower level packages and its child packages.

ASIS is the root of the interface.

Asis.Implementation

Asis.Ada_Environments maps the compilers Ada program environment to the ASIS environment.

Asis.Compilation_Units gives black box view of compilation units

Asis.Declarations, ... give white box view

Asis_Vendor_Primitives specifies vendor-specific aspects. With’s various compiler components.

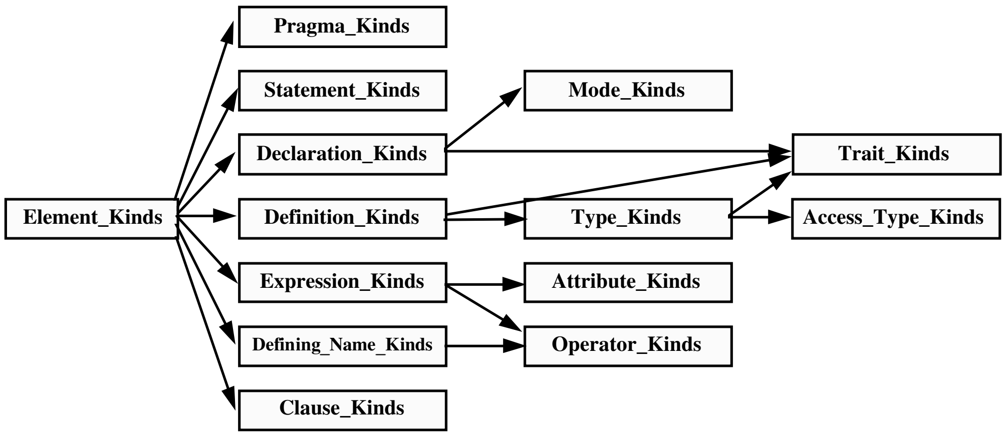
Asis_Type_Primitives specifies vendor-specific type declarations

Asis_Element_Kinds defines types for white box view

Asis_Unit_Kinds defines the compilation unit kinds

Asis_Errors defines the error kinds

Asis_Element_Kinds



These a (some of) the various enumeration types providing the classification of elements.

For an element you can query about the element kind and then use further queries according to the result.

Asis_Unit_Kinds has a similar but simpler classification scheme for compilation unit kinds.

Mode: in/out/inout/default-in

Trait: private/limited/abstract/tagged + combinations

Type: derived/enumeration/modular/access/...

Asis.Ada_Environments

- *Context* represents ASIS view of Ada environment: a set of compilation units
- Also support for source-based libraries
- Operations:

```
procedure Associate( Context    : in out Asis.Context;
                     Name       : in     Asis.Asis_String;
                     Parameters : in     Asis.Asis_String );
function Exists( Context : in Asis.Context ) return Boolean;
procedure Open( Context : in out Asis.Context );
```

- Further operations: `Close, Dissociate, Is_Open, Name, Default_Name, Default_Parameters, ...`

Jesper Jørgensen, DDC-I 16-06-96 5

Child unit Containers define operations that can further limit the scope of units to look at. Can correspond to single vendor libraries, for example.

Asis.Compilation_Units

- Black box view of units
  - » Attributes: kind, text file name, can-be-main
  - » Relationships: supporters, closure, ...
- Operations:

```
function Library_Unit_Declaration( Name    : String;
                                   Context : Asis.Context )
                                 return Asis.Compilation_Unit;
function Corresponding_Body( Library_Item : Asis.Compilation_Unit )
                           return Asis.Compilation_Unit;
```

- Further operations: `Has_Attribute, Semantic_Dependence_Order, Elaboration_Order, Subunits, Corresponding_Children, Text_Name, Unit_Kind, ...`

Queries like Library_Unit_Declaration constitute the gateway between environments and elements. Here we come from context to compilation_unit.

Relationship: closure: supporters, direct supporters, dependents, ancestors, descendants.

Asis.Elements

- White box view of units
- Queries that operate on all elements
- Operations:

```
function Unit_Declaration( Unit : Asis.Compilation_Unit )
                         return Asis.Declaration;
function Declaration_Kind( Element : Asis.Declaration )
                         return Asis.Declaration_Kinds;
generic
   type State_Information is limited private;
   with procedure Pre_Operation( ... );
   with procedure Post_Operation( ... );
procedure Traverse_Element( Element : in     Asis.Element;
                            Control : in out Traverse_Control;
                            State   : in out State_Information );
```

- Further operations: `Element_Kind, Type_Kind, Enclosing_Element, ...`

Actually this is just the most “general” of the packages providing the white box view; the others are Asis.Declarations, Asis.Expressions, ...

Here we have the basic means of coming from a compilation unit to the elements inside it.

We have the most general element traverser in Traverse_Element.

Traverse_Control: Continue, Abandon_Children, Abandon_Siblinfs, Terminate_Immediately.

Asis.Declarations

- Querying declarations
- Syntactically:

```
function Names( Declaration : Asis.Declaration )
              return Asis.Defining_Name_List;
function Initial_Value( Declaration : Asis.Declaration )
                      return Asis.Expression;
function Parameter_Profile( Declaration : Asis.Declaration )
                          return Asis.Parameter_Specification_List;
```

- Semantically:

```
function Corresponding_Type_Declaration( Decl : Asis.Declaration )
                                       return Asis.Declaration;
```


This package is just taken as an example of the various packages providing the white box view. The others are:

- Asis.Clauses,
- Asis.Definitions,
- Asis.Expressions,
- Asis.Statements.

Asis.Text

- Access to the text of ASIS elements
- Operations:

```
function Element_Span( Element : Asis.Element ) return Asis.Span;
function Line_Image( Line : Asis.Line ) return Asis.Asis_String;
function Element_Image( Line : Asis.Element )
                        return Asis.Asis_String;
```


This package does not assume (in its interface) any knowledge of the existence, location or form of the program text.

Span = ((line,column), (line,column))

Asis_Errors

- Defines kinds of errors
  - » Inappropriate_Element, Inappropriate_Line, ...
- All errors reported by raising an exception
- Status and Diagnosis values supplied to indicate cause of error
  - » Initialization_Error, Name_Error, ...

The mentioned “Inappropriate_...” are exceptions.

The mentioned “..._Error” are status values.

Diagnosis: a string.