ASIS-for-GNAT Reference Manual

The GNAT Ada Compiler
GNAT GPL Edition, Version 2015
Configuration level: 234695
Date: 2014/12/09

AdaCore

About This Manual

This Manual contains reference material for developers using ASIS-for-GNAT — GNAT's implementation of the Ada Semantic Interface Specification (ASIS). It provides information about ASIS-for-GNAT's implementation-specific[1] characteristics and current implementation limitations.

GNAT implements both Ada 95 and Ada 2005. As of January 2008, the ASIS standard is specific to Ada 95 and has not yet been updated to Ada 2005. Notwithstanding the status of the ASIS standard, ASIS-for-GNAT includes extensions that account for the new Ada 2005 functionality. You can therefore use ASIS-for-GNAT for Ada 2005 programs, keeping in mind that the Ada 2005-specific support may subsequently change as work on updating the ASIS standard proceeds.

For further information on ASIS-for-GNAT and Ada 2005, please refer to the auxiliary documents '`asis-2005-transition.txt`' and '`features-asis2005`' in the ASIS source directory.

ASIS-for-GNAT also supports some of the language features proposed for the next language revision (Ada 2015), for more information please refer to the auxiliary document '`features-asis2015`' in the ASIS source directory.

ASIS has been designed as a portable basis for many kinds of Ada code analysis tools. However, for situations where a developer may need to exploit the characteristics of a particular Ada compiler, ASIS also contains a number of implementation-specific features. These allow interfacing with the underlying Ada implementation, as well as exploiting the implementation permissions for particular queries.

Of course, any ASIS application that uses implementation-specific features may be nonportable. You should follow good programming practice and isolate and clearly document any sections of your program that make use of such features in a nonportable manner.

What This Manual Contains

This manual contains the following chapters:

- Chapter 1 [ASIS-for-GNAT and the ASIS Standard], page 3, describes the relationship between ASIS-for-GNAT and the existing ASIS International Standard.
- Chapter 2 [ASIS Extensions], page 5, describes the contents of the packages `Asis.Extensions`, `Asis.Extensions.Flat_Kinds` and `Asis.Extensions.Iterator`.

[1] The term "implementation-specific" in ASIS means what is called "implementation-defined" in the Ada Reference Manual.

- Chapter 3 [Implementation-Specific Features and Implementation Permissions], page 7, presents the aspects of the ASIS definition that are implementation specific and describes their treatment in ASIS-for-GNAT.
- Chapter 4 [Debugging Information], page 21, describes the kinds of debugging information that you can generate with ASIS-for-GNAT.

What You Should Know Before Reading This Manual

This Reference Manual assumes that you are familiar with Ada 95 language as defined by the *International Standard ISO/IEC-8652:1995*, and with ASIS 95 as defined by the *ASIS 95 International Standard ISO/IEC 15291:1999*.

This Manual supplements the information presented in the *ASIS-for-GNAT User's Guide* and uses the terminology introduced there.

Related Information

For more information, please refer to the following documents:

- *GNAT User's Guide*
- *ASIS-for-GNAT User's Guide*
- *Ada 95 Reference Manual*
- *ASIS 95 Standard*

1 ASIS-for-GNAT and the ASIS Standard

ASIS-for-GNAT implements ASIS 95[1] and contains several extensions (see Chapter 2 [ASIS Extensions], page 5) as allowed by the *ASIS Standard*, Section 1.1.3.1.

ASIS-for-GNAT declares all of the required[2] ASIS interface packages defined in the ASIS Standard. The only differences between the GNAT and the standard ASIS versions of the packages are that GNAT-for-ASIS:

- includes GNAT-specific comment headers at the beginning of each source file;
- supplies additional context clauses;
- defines the packages' private parts;
- is formatted to comply with GNAT coding style;
- declares the `Is_Dispatching_Operation` query in `Asis.Declarations`. rather than in `Asis.Expressions`. This query has `A_Declaration Element` as its argument and, according to the general principles of the ASIS package hierarchy, it should be in the `Asis.Declarations` spec;
- includes extensions that support features introduced in Ada 2005.

[1] If a query raises the `ASIS_Failed` exception with `Not_Implemented_Error` error status, this means that some part of the functionality of the query is not implemented yet. If you encounter such a situation, report it as an ordinary ASIS-for-GNAT bug. Our goal is to have the full implementation of ASIS 95 conforming to the ASIS Standard.

[2] For the optional Data Decomposition Annex, the package `Asis.Data_Decomposition.Portable_Transfer` is not provided

2 ASIS Extensions

ASIS-for-GNAT provides some additional types and queries as ASIS extensions. All these queries are defined and documented in the hierarchy headed by package `Asis.Extensions`. They are referred as "ASIS extensions" or "ASIS extension queries" below.

All the ASIS extensions obey the general ASIS rules:

- When using ASIS extensions, you have to follow the required sequencing of calls
- Only ASIS-defined exceptions propagate outside ASIS extension queries

If the documentation of an ASIS extension query contains a list of "appropriate" `Element` kinds, then the query can be applied only to `Element`s from this list, and it raises `ASIS_Inappropriate_Element` with `Value_Error` status otherwise. If the documentation of an ASIS extension query contains a list of "expected" element kinds, then the query can be applied to an `Element` having any kind, but it returns a meaningful result only for `Element`s from this list.

The current set of ASIS extensions originated from the ASIS implementation needs and from the development of some ASIS tools inside the ASIS-for-GNAT team. The `Asis.Extensions` hierarchy is not necessarily frozen: some further extension queries may be added, and suggestions from ASIS application developers are welcome.

Note that some of the ASIS extensions are implemented as ASIS *secondary queries* — that is, the implementation of such a query is a sequence of primary ASIS queries. Some other extensions are *pure extensions*; that is, their implementation is based on direct access to GNAT's internal data structures.

2.1 `Asis.Extensions`

This package, whose spec is located in the file '`asis-extensions.ads`', contains the declarations of various ASIS extensions, including dynamic `Element` and `Compilation_Unit` list types, placeholder actual parameters for `Asis.Iterator.Traverse_Element`, additional `Element` structural and semantic queries, queries that return information about the status of the source file for a `Compilation_Unit`, queries returning the (images of the) values of static expressions, etc.

2.2 `Asis.Extensions.Flat_Kinds`

The ASIS `Element` classification hierarchy is based on a set of Ada enumeration types, each corresponding to a "level" in the hierarchy. The package `Asis.Extensions.Flat_Kinds`, whose spec is located in the file '`asis-extensions-flat_kinds.ads`', defines the enumeration type

`Flat_Element_Kinds`; this type combines the values of all these types and thus provides a "flat" view onto the syntactic `Element` classification.

2.3 `Asis.Extensions.Iterator`

This package, whose spec is located in the file '`asis-extensions-iterator.ads`', contains the declarations of `Traverse_Unit` generic procedure that is a generalization of the standard ASIS `Asis.Iterator.Traverse_Element` iterator. `Traverse_Unit` provides the depth-first traversal of the whole syntactical structure of the ASIS Compilation Unit.

3 Implementation-Specific Features and Implementation Permissions

ASIS permits four kinds of implementation-specific behavior.

First, ASIS subprograms that define an interface between an ASIS implementation and the underlying Ada implementation have implementation-specific parameters. There are three such queries — `Asis.Implementation.Initialize`, `Asis.Implementation.Finalize` and `Asis.Ada_Environments.Associate`. Each has a string parameter named `Parameters` with an implementation-specific meaning. The meaning of the `Parameters` string in ASIS-for-GNAT is discussed in Section 3.1 [Interacting with the Underlying Ada Implementation], page 7.

Second, in some areas the ASIS standard explicitly grants the implementation permission to provide restricted functionality; generally this allows omitting features that could present considerable implementation difficulty. Such permissions usually affect more than one ASIS query. The ASIS package `Asis.Implementation.Permissions` contains boolean queries identifying the choices made by a given ASIS implementation. The ASIS-for-GNAT approach to these implementation permissions is discussed in Section 3.2 [Implementation Permissions], page 11.

Third, the ASIS standard defines specific implementation permissions for some queries. Also, the result of a query may be implementation specific because of the nature of the query. See Section 3.3 [ASIS Queries Having Specific Implementation Permissions or Implementation-Specific Results], page 13.

Finally, ASIS-for-GNAT provides special `Context` manipulation mechanisms that supplement those defined in the ASIS standard. These additional `Context` modes may be useful for some ASIS applications.

3.1 Interacting with the Underlying Ada Implementation

This section describes how to use the `Parameters` string to pass implementation-specific information to several ASIS subprograms.

3.1.1 Format of the `Parameters` String

A `Parameters` string is passed to three ASIS subprograms: `Asis.Implementation.Initialize`, `Asis.Implementation.Finalize`, and `Asis.Ada_Environments.Associate`.

The `Parameters` string comprises substrings delimited by separators. The substrings are called *parameters* (with lower-case 'p') below. A separator is a non-empty string comprising characters from the set `{ <Space>,<LF>, <CR>}`.

There may be 0 or more parameters in a `Parameters` string, and there may be separators before the first and/or after the last parameter.

Each of the queries `Asis.Implementation.Initialize`, `Asis.Implementation.Finalize`, and `Asis.Ada_Environments.Associate` has specific rules for the format of its parameters. If some parameter is not well-formed, then either a warning message is generated or else the `ASIS_Failed` exception is raised with the `Parameter_Error` status. The descriptions below explain the situations where `ASIS_Failed` is raised.

3.1.2 Parameters of `Asis.Implementation.Initialize`

The allowed parameters for `Asis.Implementation.Initialize` are as follows:

`-d<flag>` The specific ASIS-for-GNAT debug flag named `<flag>` is set ON

`-dall` All the ASIS-for-GNAT debug flags are set ON

`-k` Keep going even if an internal implementation error is detected. When a non-ASIS exception is raised, it is replaced by raising `ASIS_Failed` with `Unhandled_Exception_Error` status (this is the only case when `Unhandled_Exception_Error` is set) and the `Diagnosis` string containing the name and the message from the non-ASIS exception originally raised

`-nbb` No bug box. Do not output to `Standard_Error` the bug box containing the description of the internal implementation bug. Implies `-k`

`-sv` Set the strong GNAT/ASIS version check when reading the tree files

`-wv` Set the weak GNAT/ASIS version check when reading the tree files

`-we` All ASIS warnings are treated as errors. When execution reaches the point where the warning would occur, the `ASIS_Failed` exception is raised; the warning message is the ASIS `Diagnosis` string.

`-ws` All ASIS warning messages are suppressed.

The `<flag>` value for the '`-d`' parameter may be any lower case letter from `a` through `z` or any digit from `0` through `9`, although not all of the 36 possible flags are implemented. For more information, refer to the documentation in the source file '`a4g-a_debug.adb`'. See also Section 4.2 [ASIS Debug Flags], page 22.

If more then one parameter controlling the warning mode is set in the `Parameters` string, all but the last one are ignored.

3.1.3 Parameters of `Asis.Implementation.Finalize`

No parameters are allowed for `Asis.Implementation.Finalize`.

`Asis.Implementation.Finalize` resets all the general ASIS-for-GNAT parameters to their default values (that is, all the debug flags are set OFF, and the warning mode is set to the default warning mode).

3.1.4 Parameters of `Asis.Ada_Environments.Associate`

The following parameters are allowed:

`-C1` The `Context` comprises a single tree file, whose name is given as the next parameter in the `Parameters` string.

`-CN` The `Context` comprises a set of one or more tree files, whose names are given as the next set of parameters in the `Parameters` string.

`-CA` The `Context` comprises all the tree files in the tree search path.

`-FS` All the trees considered as making up a given `Context` are created "on the fly", whether or not the corresponding tree file already exists. Once created, a tree file then is reused as long as the `Context` remains open.

`-FT` Only pre-created trees are used; no tree files are created by ASIS.

`-FM` Mixed approach: if a needed tree does not exist, an attempt is made to create it "on the fly".

`-SA` Source files for all the `Compilation_Unit`s belonging to the `Context` (except the predefined `Standard` package) are considered in the consistency check when opening the `Context`.

`-SE` Only existing source files for all the `Compilation_Units` belonging to the `Context` are considered in the consistency check when opening the `Context`.

`-SN` No source files from the underlying file system are taken into account when checking the consistency of the set of tree files making up the `Context`.

`-I<dir>` Defines the directory in which to search for source files when compiling sources to create a tree "on the fly".

`--GCC=`*`compiler_name`*
Defines the program to be called to create the tree on the fly

`-gnatec<file>`
Defines the additional configuration file to be used when calling GNAT to create the tree on the fly for '`-FS`' or '`-FM`' Context

`-gnatA` Avoid processing '`gnat.adc`' when calling GNAT to create the tree on the fly for '`-FS`' or '`-FM`' Context

`-T<dir>` Defines the directory in which to search for a tree file.

`<file_name>`
Defines the name of a tree file (used in conjunction with '`-C1`' or '`-CN`').

For the '`-I`' and '`-T`' parameters, `<dir>` should denote an existing directory in the underlying file system. The "." and ".." notations are allowed, as well as relative or absolute directory names. If `<dir>` does not denote an existing directory, `ASIS_Failed` with `Parameter_Error` status is raised.

For ASIS '`-FS`' or '`-FM`' Context, Context parameters '`-I`', '`-gnatec`' and '`-gnatA`' are passed to the GNAT call to create the tree on the fly and these parameters have exactly the same meaning as they have for GNAT.

A tree file name given by a `<file_name>` parameter may or may not contain directory information.

Any relative directory name or file name containing relative directory information should start from "." or "..".

If a directory or a file name used as a part of some Context parameter contains space characters, this name should be quoted.

The search path associated with an ASIS `Context` consists of the directories listed as parameters for the `Asis.Ada_Environments.Associate` query, in the same order as they are included in the actual `Parameters` string. The ASIS source search path consists only of the directories following '`-I`', and the ASIS tree search path consists only of the directories following '`-T`'. If no source (tree) directories are present in the value of the `Parameters` string, then the ASIS source (tree) search path consists of the current directory only. Otherwise the current directory is included in the ASIS search path if and only if it is set explicitly as '`-I.`' or '`-T.`' respectively.

If an ASIS `Context` is associated with an '`-FS`' or '`-FM`' option, the `Context` source search path is used to locate sources of the units for which tree files need to be created, and to locate other source files needed during compilation. For example, if we have:

```
Asis.Ada_Environments.Associate
  (My_Context,
  "My_Context_Name",
  "-CA -FS -I./dir -I.");
```

then, when processing a call:

```
My_Unit := Asis.Compilation_Units.Library_Unit_Declaration
             ("Foo", My_Context);
```

ASIS first tries to locate the source file '`foo.ads`' in '`./dir`', and if this attempt fails, it tries to locate it in the current directory. If there is no such file in

the current directory, ASIS continues the search by looking into the directories listed in the value of `ADA_INCLUDE_PATH` environment variable. If the source file is found (say in the current directory), ASIS creates the tree file by calling the compiler:

```
$ gcc -c -gnatc -gnatt -I./dir -I. -I- foo.ads
```

If an ASIS `Context` is associated with '`-CA`' option, then, when this `Context` is opened, ASIS processes all the tree files located in the tree search path associated with the `Context`.

The following further rules define the required combinations of parameters in the actual `Parameters` string:

- '`-C1`' and '`-CN`' require '`-FT`'
- '`-FS`' and '`-FM`' require '`-SA`'

In case an incompatible combination is set, `ASIS_Failed` with `Parameter_Error` status is raised.

If the actual `Parameters` string passed to `Associate` contains no parameters, the default parameters are '`-CA`', '`-FT`', and '`-SA`'.

The '`-FS`' and '`-FM`' options define *dynamic Context modes*; they allow the content of a `Context` (that is, the set of ASIS `Compilation_Unit`s contained in the `Context`) to be changed while the `Context` is open. See Section 3.6 [Dynamic Context Modes], page 18 for more details.

For the `Name` parameter of the `Asis.Ada_Environments.Associate` query, any string can be passed as an actual parameter. No verification is performed on the contents, and no semantics are associated with this parameter.

3.2 Implementation Permissions

This section describes how ASIS-for-GNAT deals with implementation permissions.

3.2.1 `Asis.Implementation.Permissions` Queries

The Boolean queries defined in the `Asis.Implementation.Permissions` package return the following results:

Query	*Value*
`Is_Formal_Parameter_Named_Notation_Supported`	`True`
`Default_In_Mode_Supported`	`True`
`Generic_Actual_Part_Normalized`	`False`
`Record_Component_Associations_Normalized`	`False`
`Is_Prefix_Call_Supported`	`True`
`Function_Call_Parameters_Normalized`	`False`
`Call_Statement_Parameters_Normalized`	`False`
`Discriminant_Associations_Normalized`	`False`

`Is_Line_Number_Supported`	`True`
`Is_Span_Column_Position_Supported`	`True`
`Is_Commentary_Supported`	`True`
`Attributes_Are_Supported`	`False`
`Implicit_Components_Supported`	`False` (*)
`Object_Declarations_Normalized`	`False`
`Predefined_Operations_Supported`	`False` (*)
`Inherited_Declarations_Supported`	`True` (*)
`Inherited_Subprograms_Supported`	`True` (*)
`Generic_Macro_Expansion_Supported`	`True`

(*) See also Section 3.2.2 [Processing Implicit Elements], page 12.

3.2.2 Processing Implicit `Element`s

ASIS `Element`s represent both explicit and implicit[1] components of Ada programs. Some ASIS queries can return implicit `Element`s (that is, `Element`s representing implicit Ada constructs). Any syntactic or semantic query should accept an implicit `Element` as an `Element` parameter, but the ASIS Standard allows an implementation not to support implicit `Element`s at all, or to support them only partially. If an implementation does not support the implicit `Element` representing a particular kind of construct, then an ASIS query that is supposed to process this implicit `Element` should return either a `Nil_Element` or a `Nil_Element_List` depending on whether the query returns a single `Element` or an `Element_List`.

Implicit `Element`s are partially supported by ASIS-for-GNAT.

ASIS-for-GNAT supports implicit `Element`s for the following constructs:

- Derived user-defined subprograms
- Derived enumeration literals
- Derived record components

ASIS-for-GNAT does not support implicit `Element`s representing implicit declarations of predefined type operations (such as "=", or the "+" operation for numeric types).

3.2.3 Processing Several Contexts at a Time

According to the ASIS Standard, the number of ASIS `Context`s that can be associated and opened at a time, as well as the number of ASIS `Compilation_Unit`s that can be processed at a time, are implementation specific. ASIS-for-GNAT does not impose any restriction on the number of `Context`s opened at the same time, or on the number of `Compilation_Unit`s that can be obtained from

[1] An example of an implicit construct is a derived subprogram.

all the opened `Context`s, as long as the application does not go beyond general system resource limitations.

However, for a `Context` associated with an '`-FS`' or '`-FM`' option, all the trees created "on the fly" while obtaining `Compilation_Unit`s from this `Context` are placed in the current directory. If the current directory also contains some tree files belonging to another `Context`, the latter may become corrupted. To process more than one `Context` safely, an application should have at most one `Context` associated with the '`-FS`' or '`-FM`' option. Moreover, if among `Context`s processed at the same time there is one that can create trees "on the fly", then the other `Context`s should not use tree files located in the current directory.

3.2.4 Implementation-Defined Types and Values

All the implementation-defined types, subtypes and values depend on the subtype `Implementation_Defined_Integer_Type` and on the `Implementation_Defined_Integer_Constant` defined in package `Asis`. ASIS-for-GNAT's declarations for these entities are the same as in the ASIS Standard:

```
subtype Implementation_Defined_Integer_Type is Integer;
Implementation_Defined_Integer_Constant : constant := 2**31-1;
```

All the ASIS (sub)types used as list indexes for ASIS array types have `Implementation_Defined_Integer_Constant` as an upper bound.

3.3 ASIS Queries Having Specific Implementation Permissions or Implementation-Specific Results

This section documents queries having implementation permissions (given under `--|IP` sentinel in the ASIS definition) and queries whose behavior is otherwise implementation specific. Such queries are presented below in their order of appearance in the ASIS Standard. The clause and subclause numbers shown are those from the ASIS Standard.

The results returned by the ASIS `Debug_Image` queries are discussed in Section 4.1 [Interpreting Debug Images], page 21.

ASIS 8 `package Asis.Ada_Environments`

ASIS 8.1 `function Default_Name`

- Null string is returned.

ASIS 8.2 `function Default_Parameters`

- Null string is returned;.

ASIS 8.4 `procedure Open`

- For a `Context` associated with the '`-CA`' option:
  - If '`-FS`' is also set, nothing is done.

  - If the '-FT' or '-FM' is set, all the tree files (that is, files having '.adt' suffix) in the tree search path associated with the Context are processed. ASIS reads in each tree file and checks that it was created with the '-gnatc' option. Tree files that cannot be read in or that were not created with the '-gnatc' option are ignored. For each other tree ASIS collects some "black-box" information about the Compilation_Units that it represents, and performs a consistency check for every unit it encounters in the tree (see *ASIS-for-GNAT User's Guide* for a discussion of the consistency problem). If any consistency check fails, ASIS_Failed is raised and the Context remains closed.
- For a Context associated with a '-C1' or '-CN' option, ASIS processes all the tree files associated with the Context, collecting "black-box" information and performing consistency checks for all the encountered Compilation Units. If for any reason a tree file cannot be successfully read in for a Context associated with a '-C1' option, ASIS_Failed is raised and the Context remains closed. If a tree read fails for a Context associated with a '-CN' option, an ASIS warning is generated and the Context opening process continues. If any consistency check fails, ASIS_Failed is raised and the Context remains closed.

ASIS 9 package Asis.Ada_Environments.Containers

- ASIS-for-GNAT supports the trivial Container model. Every Context contains exactly one Container, whose content and name are the same as its enclosing Context

ASIS 10 package Asis.Compilation_Units

ASIS 10.3 function Unit_Origin

- A_Predefined_Unit origin is returned for those compilation units listed in RM95, Annex A(2), and only for these units.
- An_Implementation_Unit origin is returned for compilation units that are the components of the GNAT Run-Time Library, but that are not listed in RM95, Annex A(2).
- An_Application_Unit origin is returned for all other compilation units.

ASIS 10.6 function Library_Unit_Declaration and *ASIS 10.7* function Compilation_Unit_Body

- When processing a Context associated with an '-FS' or '-FM' option, if ASIS cannot find a needed unit in the tree files that have been already processed, it tries to create the needed tree by locating the source of the unit and compiling it "on the fly". If this attempt fails for any reason, Nil_Compilation_Unit is returned.

ASIS 10.13 `function Corresponding_Declaration`

- ASIS-for-GNAT does not make use of ASIS `Compilation_Unit`s of `An_Unknown_Unit` kind.
- If an argument is of `A_Public_Declaration_And_Body` class, `Nil_Compilation_Unit` is returned.

ASIS 10.14 `function Corresponding_Body`

- ASIS-for-GNAT does not make use of ASIS `Compilation_Unit`s of `An_Unknown_Unit` kind.

ASIS 10.22 `function Can_Be_Main_Program`

- For GNAT, any parameterless library procedure and any parameterless library function returning a result of an integer type is classified by this query as a (possible) main subprogram for a partition.
- If for such a library subprogram both spec and body exist as ASIS `Compilation_Unit`s retrievable from a given ASIS `Context`, both are considered as `Can_Be_Main_Program`.

ASIS 10.24 `function Text_Name`

- This function returns the name of the source file containing the source of `Compilation_Unit`. This name may or may not contain a prefix denoting the directory in the underlying file system. If present, the directory may be given in absolute or relative form, depending on the command line options that were used for the call to GNAT that created the corresponding tree file.
- This function does not check the existence of the corresponding source file in the underlying file system, it just reflects the situation which was in effect when the corresponding tree file was created. Thus, if you delete or move the corresponding source file after creating the tree, the full file name returned by this function will be incorrect.
- Use the query `Asis.Extensions.Source_File_Status` to get the information about the current status of the source file for a `Compilation_Unit`.

ASIS 10.25 `function Text_Form`

- In the GNAT compilation model all source files are ordinary text files in the underlying file system. Therefore this function always returns a `Nil_Asis_String` to indicate that `Text_IO.Open` uses the default options for manipulating Ada sources.

ASIS 10.26 `function Object_Name`

- Returns a null string. In the GNAT environment, creating an object file has no connection with creating trees for ASIS.

ASIS 10.27 `function Object_Form`

- Returns a null string.

ASIS 10.29 `function Has_Attribute`

- Returns `False`. ASIS-for-GNAT does not provide any additional attributes for Compilation Units.

ASIS 10.30 `function Attribute_Value_Delimiter`

- Returns a wide string of length one containing the `LF` wide character.

ASIS 10.31 `function Attribute_Values`

- A null string is returned.

ASIS 11 `package Asis.Compilation_Units.Times`

ASIS 11.2 `function Time_Of_Last_Update`

- This function returns the time stamp (the time of the latest change) of the corresponding source file. The corresponding source file is the source file whose name is returned by `Asis.Compilation_Units.Text_Name`.

ASIS 11.3 `function Compilation_CPU_Duration`

- This function always returns zero duration, because the CPU compilation duration concept does not apply to ASIS-for-GNAT

ASIS 11.4 `function Attribute_Time`

- This function always returns `Nil_ASIS_Time` because ASIS-for-GNAT does not provide any `Compilation_Unit` attributes

ASIS 13 `package Asis.Elements`

ASIS 13.3 `function Context_Clause_Elements`

- This function returns exactly those clauses and pragmas that are in the source for the unit.
- Returns `Nil_Element_List` if the argument unit is of `A_Nonexistent_Declaration`, `A_Nonexistent_Body` or `An_Unknown_Unit` kind
- Returns `Nil_Element_List` for the predefined package `Standard`. For all other predefined Ada compilation units, returns their context clauses as they appear in the sources held in the GNAT Run-Time Library.

ASIS 13.4 `function Configuration_Pragmas`

- This function always returns `Nil_Element_List`, because in the GNAT compilation environment “a list of pragmas that apply to all future compilation_unit elements compiled into `The_Context`” essentially depends on the GNAT options set when compiling a unit (in particular the ‘`-gnatA`’ and ‘`-gnatec`’ options), and this cannot be determined from the content of the given `Context`.

ASIS 13.5 `function Compilation_Pragmas`

- If the argument unit has been compiled on its own to produce a corresponding tree file, then the result contains the configuration pragmas from the GNAT configuration file(s) involved in this compilation. Otherwise (that is, if the argument unit has been compiled only as an effect of compiling some other unit), the result contains only those pragmas that belong to the unit's source file.
- A pragma that appears in the unit's context clause is included in the result list only if it is a configuration pragma.
- Returns `Nil_Element_List` for the predefined package `Standard`.

ASIS 13.31 `function Is_Equal`

- Two elements representing configuration pragmas belonging to `A_Configuration_Compilation` unit (or components thereof) are considered as being equal only if they are created by the same compilation (belong to the same tree).

ASIS 13.36 `function Enclosing_Element`

- ASIS-for-GNAT does not require the `Element_Context` parameter. The `Enclosing_Element` function with two parameters just calls the `Enclosing_Element` function with one parameter for its `Element` parameter.

ASIS 15 `package Asis.Declarations`

ASIS 15.24 `function Body_Block_Statement`

- If the body passed as the actual parameter has no declarative items of its own, `Asis.Statements.Is_Declare_Block` returns `False`.

ASIS 18 `package Asis.Statements`

ASIS 18.14 `function Is_Declare_Block`

- If the argument represents the dummy block statement created by `Asis.Declarations.Body_Block_Statement` function, the result will be `True` if and only if the corresponding body has declarative items.

ASIS 20 `package Asis.Text`

ASIS 20.1 `type Line`

- Lines in ASIS-for-GNAT do not contain any end-of-line characters (see RM95, 2.2(2)).

ASIS 20.22 `function Delimiter_Image`

- Returns a wide string of length one, containing the `LF` wide character.

3.4 Processing of Predefined Input-Output Packages

The GNAT compiler transforms the structure of the predefined input-output packages (`Ada.Text_IO`, `Ada.Wide_Text_IO` and `Ada.Wide_Wide_Text_IO`) to optimize compilations of their clients. The documentation of `Ada.Text_IO` says:

```
-- Note: the generic subpackages of Text_IO (Integer_IO, Float_IO, Fixed_IO,
-- Modular_IO, Decimal_IO and Enumeration_IO) appear as private children in
-- GNAT. These children are with'ed automatically if they are referenced, so
-- this rearrangement is invisible to user programs, but has the advantage
-- that only the needed parts of Text_IO are processed and loaded.
```

The same happens for `Ada.Wide_Text_IO` and `Ada.Wide_Wide_Text_IO`. In this situation ASIS follows not the Ada Standard, but the actual code contained in the GNAT Run-Time Library. That is, the `Anclosing_Compilation_Unit` for an ASIS `Element` representing `Ada.Text_IO.Inreger_IO` will be not the `Compilation_Unit` that contains the whole package `Ada.Text_IO`, but the `Compilation_Unit` representing its private child as it is described above.

The `Asis.Extensions` package contains a query named `Is_Sub_Package_Implemented_As_Child_Unit` that allows to detect such private children of predefined Ada text input-output packages.

3.5 Representation clauses and '`-gnatI`' GNAT option

GNAT '`-gnatI`' allows to ignore all the representation clauses in the code being compiled. This allows to compile the code if it contains representation clauses that are illegal in the given compilation environment. ASIS can process tree files created with '`-gnatI`', and for the ASIS `Context` that is based on such trees, ASIS does not yield `Elements` that correspond to representation clauses.

Node, that you will see these representation clauses in the text images of the enclosing `Elements`, but nevertheless you will not be able to get them as subcomponents of such `Elements`.

3.6 Dynamic `Context` Modes

If an ASIS `Context` is defined with an '`-FS`' or '`-FM`' option, then ASIS may compile sources "on the fly" to obtain `Compilation_Unit`s. Thus the content of the `Context` will not necessarily remain frozen when the `Context` is open — when ASIS gets a new `Compilation_Unit`, it "adds" it to the `Context`. The '`-FS`' and '`-FM`' options are referred to as *dynamic Context modes*.

The difference between the two modes is as follows:

'`-FS`' ASIS does not take into account any existing tree file when opening a `Context`.

'`-FM`' ASIS first processes the tree files in the tree search path. If a given `Compilation_Unit` is present in the existing set of tree files, these

tree files are used; otherwise ASIS tries to locate the source of the unit and to compile it to produce a tree file.

For both '`-FS`' and '`-FM`' `Context`s, once a tree file is created it is added to the set of tree files making up the `Context` and then it is reused (without recreating it from sources again) for the queries dealing with `Compilation_Unit`s represented by this tree.

An advantage of these dynamic `Context` modes is that you do not have to create the tree files explicitly; to users of an ASIS application based on such `Context` modes the application appears to operate directly from source files. But there is also a drawback, a consequence of the fact that the content of a `Context` may change while the `Context` is open: some ASIS queries dealing with `Compilation_Unit`s or returning lists of `Compilation_Unit`s raise the `ASIS_Failed` exception (with `Use_Error` status). These queries are as follows:

```
Asis.Compilation_Units:
     Library_Unit_Declarations
     Compilation_Unit_Bodies
     Compilation_Units
     Corresponding_Children
```

Another limitation of the dynamic `Context` mode is that ASIS uses the standard GNAT naming scheme to compute the name of the source to be compiled from the name of the corresponding Ada compilation unit. That is, if the name of the source containing the code of some unit does not follow the GNAT naming scheme, then ASIS will not locate this source, and it will treat this unit as `Nil_Compilation_Unit`.

4 Debugging Information

There are two kinds of the debugging information available in ASIS-for-GNAT — debug images returned by the ASIS query `Debug_Image` (for `Context`s, `Compilation_Unit`s and `Element`s); and debug output generated by ASIS queries when the corresponding implementation debug flag is set ON during ASIS initialization (see Section 3.1.2 [Parameters of Asis.Implementation.Initialize], page 8).

4.1 Interpreting Debug Images

It is straightforward to interpret the debug images generated for the main ASIS abstractions, because most of the information directly corresponds to ASIS concepts. The following details of debug images are implementation specific.

`Context`

`Context` Id
: This is the internal `Context` Id used in the implementation data structures. This Id is assigned to a `Context` when it is associated for the first time, and it remains unchanged and unique until ASIS is finalized.

All tree files
: The number of tree files making up the given `Context`.

`Compilation_Unit`

`Compilation_Unit` Id
: This is the internal `Compilation_Unit` Id used in the implementation data structures. This Id remains unchanged and unique until the unit's enclosed `Context` is closed.

Is consistent
: `True` if the same version of the unit's source was used for all the tree files making up the enclosed unit's context, and `False` otherwise

`Element`

Node, R_Node, Node_Field_1
: Tree nodes on which the internal representation of a given `Element` is based. They are meaningful only in the tree file indicated in the `Enclosing_Tree` field of the debug image

Special Case
: Implementation-specific indication of the cases when the `Element` needs some special processing.

Obtained from the tree
: The Id and the name of the tree file from which the tree-specific fields of the internal representation of given `Element` were obtained

`Rel_Sloc`
: Indicates the (relative) position of the source text of the `Element`, counting from the beginning of the source of its enclosing compilation unit. Applies to implicit `Element`s also.

4.2 ASIS Debug Flags

ASIS provides several internal debug flags, which are described in '`a_debug.adb`'. When one or more of these flags is set, useful internal debugging information is directed to `Standard_Output`. Although this information is not always user-oriented, you may find the following debug flags helpful when you are developing an ASIS application:

`-dc`
: Outputs the content of the internal data structures for a `Context`, when the `Context` is closed and dissociated. By analyzing this information, you may map other debug information onto unit and tree Ids.

`-di`
: Turns off including the location of an `Element` into the result generated by `Debug_Image`. This may be useful if an ASIS program crashes because of some problem with ASIS structural queries (structural queries are used by `Element`'s `Debug_Image` query to compute the source location of the argument).

`-do`
: When the `Context` is opened, lists the tree files being processed, and the ones selected to represent a given `Context`

`-dt`
: Outputs a message whenever a tree file is read in. This information may be useful for analyzing and reducing the "tree swapping profile" of your application.

Appendix A GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related

matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading

or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified

Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as “Endorsements” or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled “History” in the various original documents, forming one section entitled “History”; likewise

combine any sections entitled “Acknowledgements”, and any sections entitled “Dedications”. You must delete all sections entitled “Endorsements.”

Heading 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an “aggregate”, and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify,

sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

A

B

C

D

Table of Contents