INTERNATIONAL STANDARD

ISO/IEC 15291

First edition
1999-04-15

Information technology — Programming languages — Ada Semantic Interface Specification (ASIS)

Technologies de l'information — Langages de programmation, leurs environnements et interfaces de logiciel de système — Spécification d'interface pour la sémantique Ada

Reference number
ISO/IEC 15291:1999(E)

Contents

Figures

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

International Standard ISO/IEC 15291 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee 22, *Programming languages, their environments and system software interfaces*.

Annexes A, B, C, and D of this International Standard are for information only.

Introduction

The Ada Semantic Interface Specification (ASIS) is an interface between an Ada environment (as defined by ISO/IEC 8652:1995) and any tool requiring information from it. An Ada environment includes valuable semantic and syntactic information. ASIS is an open and published callable interface which gives CASE tool and application developers access to this information. ASIS has been designed to be independent of underlying Ada environment implementations, thus supporting portability of software engineering tools while relieving tool developers from needing to understand the complexities of an Ada environment's proprietary internal representation.

Examples of tools that benefit from the ASIS interface include: automated code monitors, browsers, call tree tools, code reformators, coding standards compliance tools, correctness verifiers, debuggers, dependency tree analysis tools, design tools, document generators, metrics tools, quality assessment tools, reverse engineering tools, re-engineering tools, style checkers, test tools, timing estimators, and translators.

The word "may" as used in this International Standard consistently means "is allowed to" (or "are allowed to"). It is used only to express permission, as in the commonly occurring phrase "an implementation may"; other words (such as "can," "could" or "might") are used to express ability, possibility, capacity, or consequentiality.

The ASIS interface consists of a set of types, subtypes, and subprograms which provide a capability to query the Ada compilation environment for syntactic and semantic information. Package **Asis** is the root of the ASIS interface. It contains common types used throughout the ASIS interface. Important common abstractions include Context, Element, and Compilation_Unit. Type Context helps identify the compilation units considered to be analyzable as part of the Ada compilation environment. Type Element is an abstraction of entities within a logical Ada syntax tree. Type Compilation_Unit is an abstraction for Ada compilation units. In addition, there are two sets of enumeration types called Element Kinds and Unit Kinds. Element Kinds are a set of enumeration types providing a mapping to the Ada syntax. Unit Kinds are a set of enumeration types describing the various kinds of compilation units.

All ASIS subprogram interfaces are provided using child packages. Some child packages also contain type and subtype interfaces local to the child package.

The child package Asis.Implementation provides queries to initialize, finalize, and query the error status of the ASIS implementation. The child package Asis.Ada_Environments encapsulates a set of queries that map physical Ada compilation and program execution environments to logical ASIS environments.

The child package Asis.Compilation_Units defines queries that deal with compilation units and serves as the gateway between Compilation_Units, Elements, and Ada_Environments. The child package Asis.Compilation_Units.Times encapsulates the time related functions used within ASIS. The child package Asis.Compilation_Units.Relations encapsulates semantic relationship concepts used in ASIS.

The child package Asis.Elements defines general Element queries and queries for pragmas. It provides information on the element kinds for further semantic analysis.

The child package Asis.Iterator provides a mechanism to perform an iterative traversal of a logical syntax tree. During the syntax tree traversal, ASIS can analyze the various elements contained within the syntax tree. ASIS can provide the application additional processing via generic procedures, which are instantiated by the application. These additional processing queries

decompose as ASIS elements from the logical Ada semantic tree. Queries are provided in the child packages: Clauses, Declarations, Definitions, Expressions, and Statements.

- child package Asis.Clauses - Defines queries dealing with context clauses and representation clauses.
- child package Asis.Declarations - Defines queries dealing with Ada declarations.
- child package Asis.Definitions - Defines queries dealing with the definition portion of Ada object, type, and subtype declarations.
- child package Asis.Expressions - Defines all queries dealing with Ada expressions.
- child package Asis.Statements - Defines queries dealing with Ada statements.

The child package Asis.Text encapsulates a set of operations to access the text of ASIS elements. It defines the operations for obtaining compilation text spans, lines, and images of elements.

The child package Asis.Ids provides a mechanism to efficiently reference ASIS elements in a persistent manner.

To support portability amongst a variety of implementors' compilation environments, certain types and constants have been identified as implementation-defined.

The child package Asis.Errors defines the kinds of errors. The exceptions that can be raised across the ASIS interface are defined in the child package Asis.Exceptions.

The interface supports one optional child package and its single child package:

- child package Asis.Data_Decomposition - The interface also includes an optional capability to decompose data values using the ASIS type information and portable data stream, representing a data value of that type.

Information technology — Programming languages — Ada Semantic Interface Specification (ASIS)

1 General

1.1 Scope

The Ada Semantic Interface Specification (ASIS) is an interface between an Ada environment (as defined by ISO/IEC 8652:1995) and any tool requiring information from this environment. An Ada environment includes valuable semantic and syntactic information. ASIS is an open and published callable interface which gives CASE tool and application developers access to this information. ASIS has been designed to be independent of underlying Ada environment implementations, thus supporting portability of software engineering tools while relieving tool developers from needing to understand the complexities of an Ada environment's proprietary internal representation.

Examples of tools that benefit from the ASIS interface include: automated code monitors, browsers, call tree tools, code reformators, coding standards compliance tools, correctness verifiers, debuggers, dependency tree analysis tools, design tools, document generators, metrics tools, quality assessment tools, reverse engineering tools, re-engineering tools, safety and security tools, style checkers, test tools, timing estimators, and translators.

This International Standard specifies the form and meaning of the ASIS interface to the Ada compilation environment.

This International Standard is applicable to tools and applications needing syntactic and semantic information in the Ada compilation environment.

1.1.1 Extent

This International Standard specifies:

- The form of the ASIS interface;
- Sequencing of ASIS calls;
- The permissible variations within this International Standard, and the manner in which they are to be documented;
- Those violations of this International Standard that a conforming implementation is required to detect, and the effect of attempting to execute a program containing such violations;

This International Standard does not specify:

- Semantics of the interface in the face of simultaneous updates to the Ada compilation environment.
- Semantics of the interface for more than one thread of control.

1.1.2 Structure

This International Standard contains twenty-three clauses and four annexes.

Clause 1 is general in nature providing the scope of this International Standard, normative references, and definitions.

Clause 2 identifies the ASIS technical concepts. Here the Ada compilation environment to which ASIS interfaces is described. The concept of queries is presented. The ASIS package architecture is presented.

The packages that comprise the ASIS International Standard are provided in Clauses 3 through 23. These packages are provided in the correct compilation order and when presented in electronic format are compilable.

- Clause 3 package Asis
- Clause 4 package Asis.Errors
- Clause 5 package Asis.Exceptions
- Clause 6 package Asis.Implementation
- Clause 7 package Asis.Implementation.Permissions
- Clause 8 package Asis.Ada_Environments
- Clause 9 package Asis.Ada_Environments.Containers
- Clause 10 package Asis.Compilation_Units
- Clause 11 package Asis.Compilation_Units.Times
- Clause 12 package Asis.Compilation_Units.Relations
- Clause 13 package Asis.Elements
- Clause 14 package Asis.Iterator
- Clause 15 package Asis.Declarations
- Clause 16 package Asis.Definitions
- Clause 17 package Asis.Expressions
- Clause 18 package Asis.Statements
- Clause 19 package Asis.Clauses
- Clause 20 package Asis.Text
- Clause 21 package Asis.Ids
- Clause 22 package Asis.Data_Decomposition (optional package)
- Clause 23 package Asis.Data_Decomposition.Portable_Transfer

The following annexes are informative:

Annex A: Glossary
Annex B: ASIS Application Examples
Annex C: Miscellaneous ASIS I/O and IDL Approaches
Annex D: Rationale

The major package interfaces visible to ASIS users are identified as clauses facilitating access from the table of contents.

The ASIS interface is compilable. Consequently, Sentinels have been used to mark portions of the ASIS text with comments appropriate to an ASIS implementor and an ASIS user.

The sentinels and their meanings are:

--|ER (Element Reference) These comments mark an element kind reference which acts as a header for those queries that work on this element kind.

--|CR (Child Reference) These sentinel comments follow sentinel comments marking element references (--ER) and reference child element queries that decompose the element into its children.

--|AN (Application Note) These comments describe suggested uses, further analysis, or other notes of interest to ASIS applications.

--|IP (Implementation Permissions) These comments describe permissions given an implementor when implementing the associated type or query.

--|IR (Implementation Requirements) These comments describe additional requirements for conforming implementations.

1.1.3 Conformity with this International Standard

1.1.3.1 Implementation conformance requirements

An *ASIS implementation* includes all the hardware and software that implements the ASIS specification for a given Ada implementation and that provides the functionality required by the ASIS specification. An ASIS *implementor* is a company, institution, or other group (such as a vendor) who develops an ASIS implementation. A conforming ASIS implementation shall meet all of the following criteria:

a) The system shall support all required interfaces defined within this International Standard. These interfaces shall support the functional behavior described herein. All interfaces in the ASIS specification are required unless the interface is specifically identified as being optional. The ASIS specification defines one optional package: Asis.Data_Decomposition. Asis.Data_Decomposition has one child package, Asis.Data_Decomposition.Portable_Transfer.

b) The system may provide additional facilities not required by this International Standard. *Extensions* are non-standard facilities (e.g., other library units, non-standard children of standard ASIS library units, subprograms, etc.) which provide additional information from ASIS types, or modify the behavior of otherwise standard ASIS facilities to provide alternative or additional functionality. Nonstandard extensions shall be identified as such in the system documentation. Nonstandard extensions, when used by an application, may change the behavior of functions or facilities defined by this International Standard. The conformance document shall define an environment in which an application can be run with the behavior specified by this International Standard. In no case except package name conflicts shall such an environment require modification of a Basic Conforming or Fully Conforming ASIS Application. An implementation shall not change package specifications in this International Standard except by:

    - Adding “with” clauses, pragmas, representation specifications, comments, and allowable pragmas. Allowable pragmas are those which do not change the semantics of the interface (e.g., List, Optimize, Page).
    - Replacing instances of the words <implementation-defined> with appropriate value(s).
    - Adding or changing private parts.
    - Making any other changes that are lexically transparent to Ada compilers.

c) An ASIS implementation shall not raise Program_Error on elaboration of an ASIS package, or on execution of an ASIS subprogram, due to elaboration order dependencies in the ASIS implementation.

d) Except as explicitly provided for in this International Standard, Standard.Storage_Error is the only exception that should be raised by operations declared in this International Standard.

e) When executed, an implementation of this International Standard shall not be erroneous, as defined by ISO/IEC 8652:1995.

1.1.3.2 Implementation conformance documentation

A conformance document shall be available for an implementation claiming conformance to this International Standard. The conformance document shall have the same structure as this International Standard, with the information presented in the equivalently numbered clauses, and subclauses. The conformance document shall not contain information about extended facilities or capabilities outside the scope of this International Standard.

The conformance document shall contain a statement that indicates the full name, number, and date of the International Standard that applies. The conformance document may also list software standards approved by ISO/IEC or any ISO/IEC member body that are available for use by a Basic or Fully Conforming ASIS Application. Applicable characteristics whose documentation is required by one of these standards, or by standards of government bodies, may also be included.

The conformance document shall describe the behavior of the implementation for all implementation-defined features defined in this International Standard. This requirement shall be met by listing these features and providing either a specific reference to the system documentation or providing full syntax and semantics of these features. The conformance document shall specify the behavior of the implementation for those features where this International Standard states that implementations may vary.

No specifications other than those described in this subclause shall be present in the conformance document.

The phrase “shall be documented” in this International Standard means that documentation of the feature shall appear in the conformance document, as described previously, unless the system documentation is explicitly mentioned.

The system documentation should also contain the information found in the conformance document.

1.1.3.3 Implementation conformance categories

An implementation is required to define all of the subprograms for all of the operations defined in this International Standard, including those whose implementation is optional. *Required functionality* is the subset of ASIS facilities which are not explicitly identified in the ASIS standard as optional. *Optional functionality* is the subset of ASIS facilities which are explicitly identified in the ASIS standard as optional which may legitimately be omitted from a Basic Conforming ASIS implementation. Optional interfaces shall be included in any Fully Conforming ASIS implementation, unless stated otherwise in the ASIS specification. An application that accesses an Ada environment’s semantic tree (e.g., Diana Tree) directly using work-arounds is not considered to be a conformant application. All Conforming Applications fall within one of the categories defined below.

If an unimplemented feature is used, the exception Asis.ASIS_Failed shall be raised and Asis.Implementation_Status shall return the value for Error_Kinds of Not_Implemented_Error.

There are four categories of conforming ASIS implementations:

1.1.3.3.1 Basic conforming ASIS implementation

A Basic Conforming ASIS Implementation is an ASIS implementation supporting all required interfaces defined within this International Standard.

1.1.3.3.2 Fully conforming ASIS implementation

A Fully Conforming ASIS Implementation is an ASIS implementation supporting all required and all optional interfaces defined within this International Standard.

1.1.3.3.3 Basic conforming ASIS implementation using extensions

A Basic Conforming ASIS Implementation Using Extensions is an ASIS implementation that differs from a Basic Conforming ASIS Implementation only in that it uses nonstandard extensions that are consistent with this International Standard. Such an implementation shall fully document its extended facilities, in addition to the documentation required for a Basic Conforming ASIS Implementation.

1.1.3.3.4 Fully conforming ASIS implementation using extensions

A Fully Conforming ASIS Implementation Using Extensions is an ASIS implementation that differs from a Fully Conforming ASIS Implementation only in that it uses nonstandard extensions that are consistent with this International Standard. Such an implementation shall fully document its extended facilities, in addition to the documentation required for a Fully Conforming ASIS Implementation.

1.1.3.4 Application conformance categories

An *ASIS application* is any programming system or any set of software components making use of ASIS queries to obtain information about any set of Ada components. All ASIS applications claiming conformance to this International Standard shall use a Conforming ASIS Implementation with or without extensions.

1.1.3.4.1 Basic conforming ASIS application

A Basic Conforming ASIS Application is an application that only uses the required facilities defined within this International Standard. It shall be portable to any Conforming ASIS Implementation.

1.1.3.4.2 Fully conforming ASIS application

A Fully Conforming ASIS Application is an application that only uses the required facilities and the optional facilities defined within this International Standard. It shall be portable to any Fully Conforming ASIS Implementation.

1.1.3.4.3 Basic conforming ASIS application using extensions

A Basic Conforming ASIS Application Using Extensions is an application that differs from a Basic Conforming ASIS Application only in that it uses nonstandard, implementation provided, extended facilities that are consistent with this International Standard. Such an application should fully document its requirements for these extended facilities. A Basic Conforming ASIS Application Using Extensions may or may not be portable to other Basic or Fully Conforming ASIS Implementation Using Extensions.

1.1.3.4.4 Fully conforming ASIS application using extensions

A Fully Conforming ASIS Application Using Extensions is an application that differs from a Fully Conforming ASIS Application only in that it uses nonstandard, implementation provided, extended facilities that are consistent with this International Standard. Such an application should fully document its requirements for these extended facilities. A Fully Conforming ASIS Application Using Extensions may or may not be portable to other Fully Conforming ASIS Implementation Using Extensions.

1.1.4 Implementation permissions

The ASIS Application Program Interface (API) may be implemented through a variety of approaches. Approaches permitted by this International Standard are based on the traditional approach and the client /server approach. These implementation permissions are depicted in Figure 1 and described below:

1.1.4.1 Traditional approach (permission 1)

Traditionally, the ASIS API implementation is intended to execute on the node containing the implementor's Ada software engineering environment and the desired Ada compilation environment. Because the ASIS API interfaces directly, ASIS performs at its best. It is expected that most ASIS implementors will support this approach as it requires little additional effort when alternative approaches are supported. In Figure 1, the client tool using Permission 1 uses the ASIS specification exactly as specified in this International Standard. ASIS tools and applications are compiled in the implementor's environment.

1.1.4.2 Client / server approach (permission 2)

As an alternative, a client / server approach can be used to implement the ASIS API. Here the ASIS API is supported by a server; ASIS client tools can request ASIS services within the supported network.

Figure 1 identifies four ASIS client tools using permission 2 capable of interfacing with an ASIS Object Request Broker (ORB) server. One client tool is written in Ada, one in Java, one in C++, and one in Smalltalk. The ORB serves as a broker between the client and server on a network consisting of many nodes. Server location and services are registered with the ORB. A client needing the services interfaces with the ORB, who brokers the needed server interface information. The interface between a client and server is written as an interface specification in the Interface Definition Language (IDL). IDL is very different from most computer languages; when IDL is compiled, the interface specification is produced in either Ada, Java, C++, or Smalltalk. In addition, the necessary artifacts are produced to register the client or server interface with the ORB.

1.1.4.3 Distributed traditional approach (permission 3)

The Ada specification created by the compilation of this ASIS API in IDL is semantically equivalent to this ASIS standard, but not syntactically identical. An ASIS Client tool written in Ada interfaces to the ASIS API as specified in this International Standard. As shown in Figure 1, the ASIS API encapsulates the ASIS ORB client as generated from the compilation of the ASIS IDL into Ada. Client tools using either permission 1 or permission 3 are, most likely, identical. Client tools developed using permission 3 can be developed as plug and play.

1.1.4.4 ASIS dynamic client approach (permission 4)

In addition to using traditional compiled tools through the client / server interface, ORBs can provide a Dynamic Interface Invocation (DII) capability where rather general purpose tools can access the interface dynamically. Shown in Figure 1, such a tool behaves more like a browser. It accesses the ASIS IDL as registered with the server and browses through the services provided by the ASIS interface. Use of this capability with ASIS is extremely cumbersome and manually intensive. However, this provides a user access to information across the interface that had not been preprogrammed by a tool.

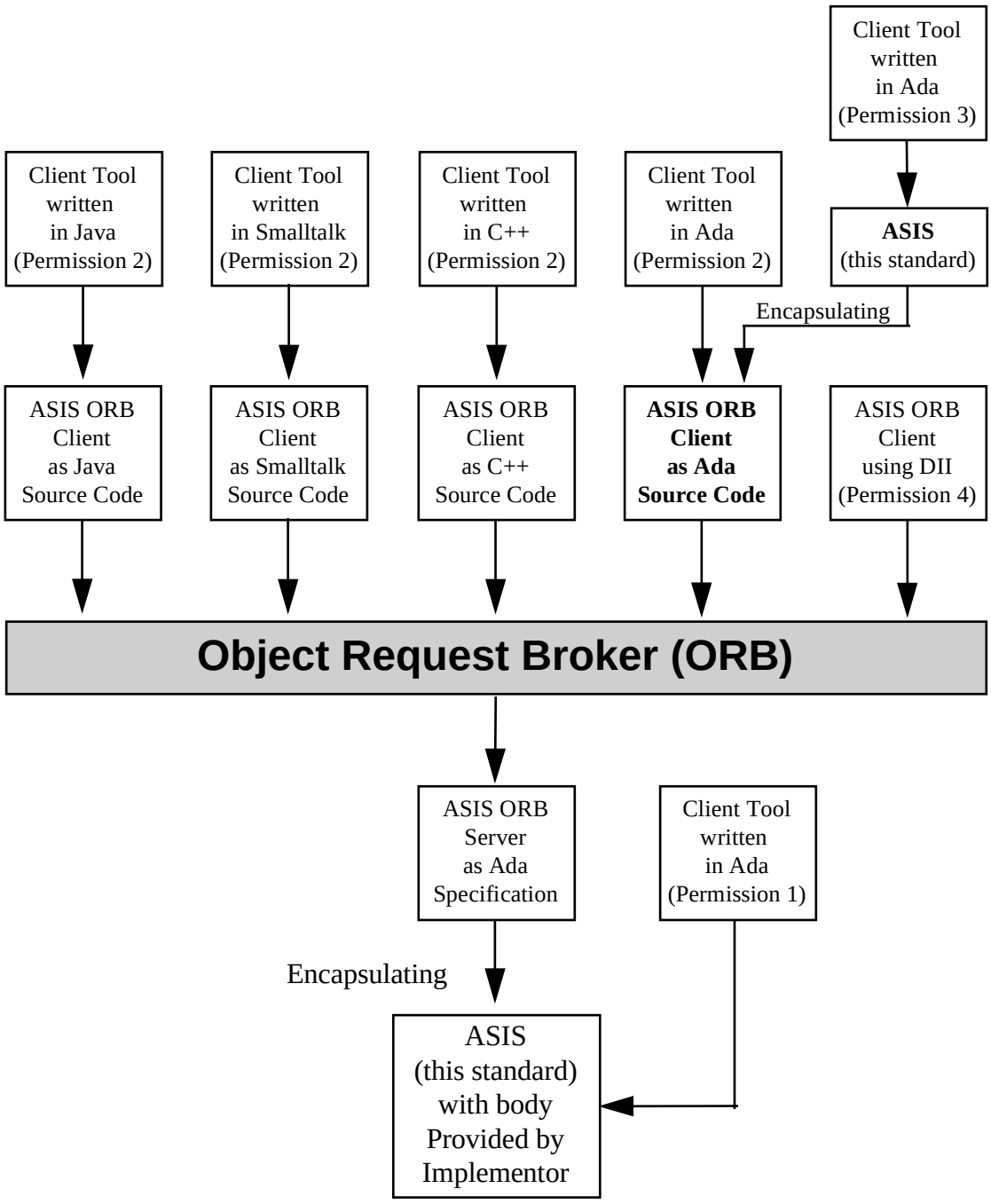

Figure 1 — ASIS implementation permissions

1.1.5 Classification of errors

ASIS reports all operational errors by raising an exception. Whenever an ASIS implementation raises one of the exceptions declared in package Asis.Exceptions, it will previously have set the values returned by the Status and Diagnosis queries to indicate the cause of the error. The possible values for Status are indicated here along with suggestions for the associated contents of the Diagnosis string.

ASIS applications are encouraged to follow this same convention whenever they explicitly raise any ASIS exception to always record a Status and Diagnosis prior to raising the exception. Values of errors along with their general meanings are:

```
Not_An_Error              -- No error is presently recorded
Value_Error               -- Routine argument value invalid
Initialization_Error      -- ASIS is uninitialized
Environment_Error         -- ASIS could not initialize
Parameter_Error           -- Bad Parameter given to Initialize
Capacity_Error            -- Implementation overloaded
Name_Error                -- Context/unit not found
Use_Error                 -- Context/unit not use/open-able
Data_Error                -- Context/unit bad/invalid/corrupt
Text_Error                -- The program text cannot be located
Storage_Error             -- Storage_Error suppressed
Obsolete_Reference_Error  -- Semantic reference is obsolete
Unhandled_Exception_Error -- Unexpected exception suppressed
Not_Implemented_Error     -- Functionality not implemented
Internal_Error            -- Implementation internal failure
```

Diagnostic messages may be more specific.

A set of exceptions shall be raised for the following circumstances:

- **ASIS_Inappropriate_Context** - Raised when ASIS is passed a Context value that is not appropriate for the operation. This exception typically indicates that a user error has occurred within the application.
- **ASIS_Inappropriate_Compilation_Unit** - Raised when ASIS is passed a Compilation_Unit value that is not appropriate. This exception typically indicates that a user error has occurred within the application.
- **ASIS_Inappropriate_Element** - Raised when ASIS is given an Element value that is not appropriate. This exception typically indicates that a user error has occurred within the application.
- **ASIS_Inappropriate_Line** - Raised when ASIS is given a Line value that is not appropriate.
- **ASIS_Inappropriate_Line_Number** - Raised when ASIS is given a Line_Number value that is not appropriate. This exception typically indicates that a user error has occurred within the application.
- **ASIS_Failed** - All ASIS routines may raise ASIS_Failed whenever they cannot normally complete their operation. This exception typically indicates a failure of the underlying ASIS implementation. This is a catch-all exception that is raised for different reasons in different ASIS implementations.

1.2 Normative reference

The following standard contains provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the International Standard indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 8652:1995, *Information technology — Programming languages — Ada.*