

This is a preview - [click here to buy the full publication](#)

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

FOREWORD	XIII
INTRODUCTION	XIV
1 GENERAL.....	1
1.1 Scope	1
1.1.1 Extent	1
1.1.2 Structure	2
1.1.3 Conformity with this International Standard	3
1.1.3.1 Implementation conformance requirements	3
1.1.3.2 Implementation conformance documentation	4
1.1.3.3 Implementation conformance categories	4
1.1.3.4 Application conformance categories	5
1.1.4 Implementation permissions	6
1.1.4.1 Traditional approach (permission 1)	6
1.1.4.2 Client / server approach (permission 2)	6
1.1.4.3 Distributed traditional approach (permission 3)	6
1.1.4.4 ASIS dynamic client approach (permission 4)	6
1.1.5 Classification of errors	8
1.2 Normative reference	9
1.3 Terms and definitions	9
2 ASIS TECHNICAL CONCEPTS.....	10
2.1 Ada compilation environment	10
2.1.1 Ada environment	10
2.1.2 ASIS notion of the Ada compilation environment	11
2.1.3 Illegal / inconsistent units in the compilation environment	12
2.2 ASIS queries	12
2.2.1 Structural queries	12
2.2.2 Semantic queries	12
2.2.3 General ASIS query processing	12
2.2.3.1 Elements and element kinds	12
2.2.3.2 Processing specific constructs	14
2.2.3.3 Element list processing	14
2.2.3.4 Operations that apply to all elements	15
2.2.3.5 Semantic references	16
2.3 ASIS package architecture	18
2.4 Application use	20
2.4.1 Establishing ASIS context	20
2.4.2 Required sequencing of calls	21
2.4.3 Notional ASIS application	21
2.4.4 Erroneous applications	24
2.4.5 Usage rules	24
2.4.5.1 General usage rules	24
2.4.5.2 Rules for processing queries for illegal/inconsistent context	25

© ISO/IEC 1999

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

3	PACKAGE ASIS	27
3.1	type ASIS_Integer	28
3.2	type ASIS_Natural	28
3.3	type ASIS_Positive	28
3.4	type List_Index	28
3.5	type Context	28
3.6	type Element	29
3.7	type Element_List	30
3.8	subtypes of Element and Element_List	30
3.9	Element Kinds	31
3.9.1	type Element_Kinds	31
3.9.2	type Pragma_Kinds	32
3.9.3	type Defining_Name_Kinds	33
3.9.4	type Declaration_Kinds	33
3.9.5	type Trait_Kinds	35
3.9.6	type Declaration_Origins	36
3.9.7	type Mode_Kinds	36
3.9.8	type Subprogram_Default_Kinds	36
3.9.9	type Definition_Kinds	36
3.9.10	type Type_Kinds	37
3.9.11	type Formal_Type_Kinds	37
3.9.12	type Access_Type_Kinds	38
3.9.13	type Root_Type_Kinds	38
3.9.14	type Constraint_Kinds	39
3.9.15	type Discrete_Range_Kinds	39
3.9.16	type Association_Kinds	39
3.9.17	type Expression_Kinds	39
3.9.18	type Operator_Kinds	40
3.9.19	type Attribute_Kinds	41
3.9.20	type Statement_Kinds	42
3.9.21	type Path_Kinds	43
3.9.22	type Clause_Kinds	44
3.9.23	type Representation_Clause_Kinds	44
3.10	type Compilation_Unit	44
3.11	type Compilation_Unit_List	45
3.12	Unit Kinds	45
3.12.1	type Unit_Kinds	45
3.12.2	type Unit_Classes	47
3.12.3	type Unit_Origins	47
3.12.4	type Relation_Kinds	47
3.13	type Traverse_Control	49
3.14	type Program_Text	49
4	PACKAGE ASIS.ERRORS	51
4.1	type Error_Kinds	51
5	PACKAGE ASIS.EXCEPTIONS	52
6	PACKAGE ASIS.IMPLEMENTATION	53
6.1	function ASIS_Version	53
6.2	function ASIS_Implementor	53
6.3	function ASIS_Implementor_Version	53
6.4	function ASIS_Implementor_Information	53
6.5	function Is_Initialized	53
6.6	procedure Initialize	53
6.7	function Is_Finalized	54
6.8	procedure Finalize	54
6.9	function Status	54

6.10	function Diagnosis	54
6.11	procedure Set_Status	55
7	PACKAGE ASIS.IMPLEMENTATION.PERMISSIONS.....	56
7.1	function Is_Formal_Parameter_Named_Notation_Supported	56
7.2	function Default_In_Mode_Supported	56
7.3	function Generic_Actual_Part_Normalized	56
7.4	function Record_Component_Associations_Normalized	56
7.5	function Is_Prefix_Call_Supported	57
7.6	function Function_Call_Parameters_Normalized	57
7.7	function Call_Statement_Parameters_Normalized	57
7.8	function Discriminant_Associations_Normalized	57
7.9	function Is_Line_Number_Supported	58
7.10	function Is_Span_Column_Position_Supported	58
7.11	function Is_Commentary_Supported	58
7.12	function Attributes_Are_Supported	58
7.13	function Implicit_Components_Supported	58
7.14	function Object_Declarations_Normalized	58
7.15	function Predefined_Operations_Supported	59
7.16	function Inherited_Declarations_Supported	59
7.17	function Inherited_Subprograms_Supported	59
7.18	function Generic_Macro_Expansion_Supported	59
8	PACKAGE ASIS.ADA_ENVIRONMENTS	60
8.1	function Default_Name	60
8.2	function Default_Parameters	60
8.3	procedure Associate	60
8.4	procedure Open	61
8.5	procedure Close	61
8.6	procedure Dissociate	61
8.7	function Is_Equal	62
8.8	function Is_Identical	62
8.9	function Exists	63
8.10	function Is_Open	63
8.11	function Has_Associations	63
8.12	function Name	63
8.13	function Parameter	64
8.14	function Debug_Image	64
9	PACKAGE ASIS.ADA_ENVIRONMENTS.CONTAINERS.....	65
9.1	type Container	65
9.2	type Container_List	65
9.3	function Defining_Containers	65
9.4	function Enclosing_Context	66
9.5	function Library_Unit_Declaration	66
9.6	function Compilation_Unit_Bodies	67
9.7	function Compilation_Units	67
9.8	function Is_Equal	67
9.9	function Is_Identical	68
9.10	function Name	68
10	PACKAGE ASIS.COMPILATION_UNITS.....	69
10.1	function Unit_Kind	69
10.2	function Unit_Class	69
10.3	function Unit_Origin	70
10.4	function Enclosing_Context	70

10.5	function Enclosing_Container	70
10.6	function Library_Unit_Declaration	71
10.7	function Compilation_Unit_Body	71
10.8	function Library_Unit_Declarations	71
10.9	function Compilation_Unit_Bodies	72
10.10	function Compilation_Units	72
10.11	function Corresponding_Children	72
10.12	function Corresponding_Parent_Declaration	73
10.13	function Corresponding_Declaration	74
10.14	function Corresponding_Body	76
10.15	function Is_Nil	77
10.16	function Is_Nil	77
10.17	function Is_Equal	77
10.18	function Is_Identical	78
10.19	function Unit_Full_Name	78
10.20	function Unique_Name	78
10.21	function Exist	79
10.22	function Can_Be_Main_Program	79
10.23	function Is_Body_Required	79
10.24	function Text_Name	79
10.25	function Text_Form	80
10.26	function Object_Name	80
10.27	function Object_Form	80
10.28	function Compilation_Command_Line_Options	81
10.29	function Has_Attribute	81
10.30	function Attribute_Value_Delimiter	81
10.31	function Attribute_Values	81
10.32	function Subunits	82
10.33	function Corresponding_Subunit_Parent_Body	82
10.34	function Debug_Image	83
11	PACKAGE ASIS.COMPILED_UNITS.TIMES	84
11.1	type Time	84
11.2	function Time_Of_Last_Update	84
11.3	function Compilation_CPU_Duration	84
11.4	function Attribute_Time	85
12	PACKAGE ASIS.COMPILED_UNITS.RELATIONS	86
12.1	type Relationship	86
12.2	constant Nil_Relationship	88
12.3	function Semantic_Dependence_Order	88
12.4	function Elaboration_Order	89
13	PACKAGE ASIS.ELEMENTS	91
13.1	function Unit_Declaration	91
13.2	function Enclosing_Compilation_Unit	91
13.3	function Context-Clause_Elements	92
13.4	function Configuration_Pragmas	92
13.5	function Compilation_Pragmas	93
13.6	function Element_Kind	94
13.7	function Pragma_Kind	94
13.8	function Defining_Name_Kind	95
13.9	function Declaration_Kind	95
13.10	function Trait_Kind	95
13.11	function Declaration_Origin	96
13.12	function Mode_Kind	96

13.13	function Default_Kind	96
13.14	function Definition_Kind	97
13.15	function Type_Kind	97
13.16	function Formal_Type_Kind	97
13.17	function Access_Type_Kind	97
13.18	function Root_Type_Kind	98
13.19	function Constraint_Kind	98
13.20	function Discrete_Range_Kind	98
13.21	function Expression_Kind	99
13.22	function Operator_Kind	99
13.23	function Attribute_Kind	99
13.24	function Association_Kind	99
13.25	function Statement_Kind	100
13.26	function Path_Kind	100
13.27	function Clause_Kind	100
13.28	function Representation_Clause_Kind	100
13.29	function Is_Nil	101
13.30	function Is_Nil	101
13.31	function Is_Equal	101
13.32	function Is_Identical	101
13.33	function Is_Part_Of_Implicit	102
13.34	function Is_Part_Of_Inherited	103
13.35	function Is_Part_Of_Instance	103
13.36	function Enclosing_Element	103
13.37	function Pragmas	105
13.38	function Corresponding_Pragmas	106
13.39	function Pragma_Name_Image	106
13.40	function Pragma_Argument_Associations	106
13.41	function Debug_Image	107
13.42	function Hash	107
14	PACKAGE ASIS.ITERATOR	108
14.1	procedure Traverse_Element	108
15	PACKAGE ASIS.DECLARATIONS	110
15.1	function Names	110
15.2	function Defining_Name_Image	111
15.3	function Position_Number_Image	111
15.4	function Representation_Value_Image	112
15.5	function Defining_Prefix	112
15.6	function Defining_Selector	112
15.7	function Discriminant_Part	113
15.8	function Type_Declaration_View	113
15.9	function Object_Declaration_View	114
15.10	function Initialization_Expression	115
15.11	function Corresponding_Constant_Declaration	115
15.12	function Declaration_Subtype_Mark	116
15.13	function Corresponding_Type_Declaration	117
15.14	function Corresponding_First_Subtype	117
15.15	function Corresponding_Last_Constraint	118
15.16	function Corresponding_Last_Subtype	118
15.17	function Corresponding_Representation_Clauses	119
15.18	function Specification_Subtype_Definition	119
15.19	function Parameter_Profile	120
15.20	function Result_Profile	120
15.21	function Body_Declarative_Items	121

15.22	function Body_Statements	121
15.23	function Body_Exception_Handlers	122
15.24	function Body_Block_Statement	122
15.25	function Is_Name_Repeated	123
15.26	function Corresponding_Declaration	123
15.27	function Corresponding_Body	125
15.28	function Corresponding_Subprogram_Derivation	127
15.29	function Corresponding_Type	127
15.30	function Corresponding_Equality_Operator	128
15.31	function Visible_Part_Declarative_Items	128
15.32	function Is_Private_Present	129
15.33	function Private_Part_Declarative_Items	129
15.34	function Renamed_Entity	130
15.35	function Corresponding_Base_Entity	130
15.36	function Protected_Operation_Items	131
15.37	function Entry_Family_Definition	132
15.38	function Entry_Index_Specification	133
15.39	function Entry_Barrier	133
15.40	function Corresponding_Subunit	133
15.41	function Is_Subunit	134
15.42	function Corresponding_Body_Stub	134
15.43	function Generic_Formal_Part	135
15.44	function Generic_Unit_Name	137
15.45	function Generic_Actual_Part	137
15.46	function Formal_Subprogram_Default	138
15.47	function Corresponding_Generic_Element	139
15.48	function Is_Dispatching_Operation	139
16	PACKAGE ASIS.DEFINITIONS.....	140
16.1	function Corresponding_Type_Operators	140
16.2	function Parent_Subtype_Indication	141
16.3	function Record_Definition	141
16.4	function Implicit_Inherited_Declarations	141
16.5	function Implicit_Inherited_Subprograms	142
16.6	function Corresponding_Parent_Subtype	143
16.7	function Corresponding_Root_Type	143
16.8	function Corresponding_Type_Structure	144
16.9	function Enumeration_Literal_Declarations	144
16.10	function Integer_Constraint	145
16.11	function Mod_Static_Expression	145
16.12	function Digits_Expression	145
16.13	function Delta_Expression	146
16.14	function Real_Range_Constraint	146
16.15	function Index_Subtype_Definitions	147
16.16	function Discrete_Subtype_Definitions	147
16.17	function Array_Component_Definition	147
16.18	function Access_To_Object_Definition	148
16.19	function Access_To_Subprogram_Parameter_Profile	148
16.20	function Access_To_Function_Result_Profile	149
16.21	function Subtype_Mark	149
16.22	function Subtype_Constraint	150
16.23	function Lower_Bound	150
16.24	function Upper_Bound	151
16.25	function Range_Attribute	151
16.26	function Discrete_Ranges	152

16.27	function Discriminant_Associations	152
16.28	function Component_Subtype_Indication	153
16.29	function Discriminants	154
16.30	function Record_Components	154
16.31	function Implicit_Components	155
16.32	function Discriminant_Direct_Name	155
16.33	function Variants	156
16.34	function Variant_Choices	156
16.35	function Ancestor_Subtype_Indication	157
16.36	function Visible_Part_Items	157
16.37	function Private_Part_Items	157
16.38	function Is_Private_Present	158
17	PACKAGE ASIS.EXPRESSIONS.....	159
17.1	function Corresponding_Expression_Type	159
17.2	function Value_Image	160
17.3	function Name_Image	160
17.4	function References	161
17.5	function Is_Referenced	161
17.6	function Corresponding_Name_Definition	162
17.7	function Corresponding_Name_Definition_List	164
17.8	function Corresponding_Name_Declaration	164
17.9	function Prefix	165
17.10	function Index_Expressions	165
17.11	function Slice_Range	166
17.12	function Selector	166
17.13	function Attribute_Designator_Identifier	166
17.14	function Attribute_Designator_Expressions	167
17.15	function Record_Component_Associations	167
17.16	function Extension_Aggregate_Expression	168
17.17	function Array_Component_Associations	168
17.18	function Array_Component_Choices	169
17.19	function Record_Component_Choices	169
17.20	function Component_Expression	170
17.21	function Formal_Parameter	170
17.22	function Actual_Parameter	171
17.23	function Discriminant_Selector_Names	172
17.24	function Discriminant_Expression	173
17.25	function Is_Normalized	173
17.26	function Is_Defaulted_Association	174
17.27	function Expression_Parenthesized	174
17.28	function Is_Prefix_Call	175
17.29	function Corresponding_Called_Function	175
17.30	function Function_Call_Parameters	176
17.31	function Short_Circuit_Operation_Left_Expression	177
17.32	function Short_Circuit_Operation_Right_Expression	177
17.33	function Membership_Test_Expression	177
17.34	function Membership_Test_Range	178
17.35	function Membership_Test_Subtype_Mark	178
17.36	function Converted_Or_Qualified_Subtype_Mark	178
17.37	function Converted_Or_Qualified_Expression	179
17.38	function Allocator_Subtype_Indication	179
17.39	function Allocator_Qualified_Expression	179

18 PACKAGE ASIS.STATEMENTS	181
18.1 function Label_Names	181
18.2 function Assignment_Variable_Name	181
18.3 function Assignment_Expression	181
18.4 function Statement_Paths	182
18.5 function Condition_Expression	182
18.6 function Sequence_Of_Statements	182
18.7 function Case_Expression	183
18.8 function Case_Statement_Alternative_Choices	183
18.9 function Statement_Identifier	183
18.10 function Is_Name_Repeated	184
18.11 function While_Condition	184
18.12 function For_Loop_Parameter_Specification	184
18.13 function Loop_Statements	185
18.14 function Is_Declare_Block	185
18.15 function Block_Declarative_Items	185
18.16 function Block_Statements	186
18.17 function Block_Exception_Handlers	186
18.18 function Exit_Loop_Name	186
18.19 function Exit_Condition	187
18.20 function Corresponding_Loop_Exited	187
18.21 function Return_Expression	187
18.22 function Goto_Label	188
18.23 function Corresponding_Destination_Statement	188
18.24 function Called_Name	188
18.25 function Corresponding_Called_Entity	189
18.26 function Call_Statement_Parameters	190
18.27 function Accept_Entry_Index	191
18.28 function Accept_Entry_Direct_Name	191
18.29 function Accept_Parameters	191
18.30 function Accept_Body_Statements	192
18.31 function Accept_Body_Exception_Handlers	192
18.32 function Corresponding_Entry	192
18.33 function Requeue_Entry_Name	193
18.34 function Delay_Expression	193
18.35 function Guard	193
18.36 function Aborted_Tasks	194
18.37 function Choice_Parameter_Specification	194
18.38 function Exception_Choices	194
18.39 function Handler_Statements	195
18.40 function Raised_Exception	195
18.41 function Qualified_Expression	195
18.42 function Is_Dispatching_Call	196
18.43 function Is_Call_On_Dispatching_Operation	196
19 PACKAGE ASIS.CLAUSES	197
19.1 function Clause_Names	197
19.2 function Representation_Clause_Name	197
19.3 function Representation_Clause_Expression	198
19.4 function Mod_Clause_Expression	198
19.5 function Component_Clauses	198
19.6 function Component_Clause_Position	199
19.7 function Component_Clause_Range	199

20 PACKAGE ASIS.TEXT	200
20.1 type Line	200
20.2 type Line_Number	200
20.3 type Line_Number_Positive	201
20.4 type Line_List	201
20.5 type Character_Position	201
20.6 type Character_Position_Positive	201
20.7 type Span	201
20.8 function First_Line_Number	202
20.9 function Last_Line_Number	202
20.10 function Element_Span	202
20.11 function Compilation_Unit_Span	203
20.12 function Compilation_Span	203
20.13 function Is_Nil	203
20.14 function Is_Nil	203
20.15 function Is_Nil	204
20.16 function Is_Equal	204
20.17 function Is_Identical	204
20.18 function Length	204
20.19 function Lines	204
20.20 function Lines	205
20.21 function Lines	206
20.22 function Delimiter_Image	206
20.23 function Element_Image	206
20.24 function Line_Image	207
20.25 function Non_Comment_Image	207
20.26 function Comment_Image	207
20.27 function Is_Text_Available	208
20.28 function Debug_Image	208
21 PACKAGE ASIS.IDS	209
21.1 type Id	209
21.2 function Hash	209
21.3 function "<"	209
21.4 function ">"	209
21.5 function Is_Nil	210
21.6 function Is_Equal	210
21.7 function Create_Id	210
21.8 function Create_Element	210
21.9 function Debug_Image	211
22 PACKAGE ASIS.DATA_DECOMPOSITION (OPTIONAL).....	212
22.1 type Record_Component	214
22.2 type Record_Component_List	214
22.3 type Array_Component	215
22.4 type Array_Component_List	215
22.5 type Dimension_Indexes	215
22.6 type Array_Component_Iterator	216
22.7 type Portable_Data	216
22.8 type Type_Model_Kinds	218
22.9 function Type_Model_Kind	218
22.10 function Is_Nil	218
22.11 function Is_Equal	218
22.12 function Is_Identical	219
22.13 function Is_Array	219

22.14	function Is_Record	219
22.15	function Done	220
22.16	procedure Next	220
22.17	procedure Reset	220
22.18	function Array_Index	220
22.19	function Array_Indexes	220
22.20	function Discriminant_Components	221
22.21	function Record_Components	221
22.22	function Record_Components	222
22.23	function Array_Components	223
22.24	function Array_Iterator	223
22.25	function Component_Data_Stream	224
22.26	function Component_Declaration	224
22.27	function Component_Indication	225
22.28	function All_Named_Components	225
22.29	function Array_Length	225
22.30	function Array_Length	226
22.31	function Size	226
22.32	function Size	226
22.33	function Position	227
22.34	function First_Bit	227
22.35	function Last_Bit	228
22.36	function Portable_Constrained_Subtype	228
22.37	function Construct_Artificial_Data_Stream	229
23 PACKAGE ASIS.DATA_DECOMPOSITION.PORTABLE_TRANSFER		231
23.1	generic package Portable_Constrained_Subtype	231
23.2	generic package Portable_Unconstrained_Record_Type	231
23.3	generic package Portable_Array_Type_1	232
23.4	generic package Portable_Array_Type_2	232
23.5	generic package Portable_Array_Type_3	232
 ANNEXES		
A GLOSSARY		234
B ASIS APPLICATION EXAMPLES		237
B.1	Use to traverse compilation unit	237
B.2	Use to build call tree	240
C MISCELLANEOUS ASIS I/O AND IDL APPROACHES.....		244
C.1	package Portable_Data_Io	244
C.2	package Asis.Ids.Id_Io	247
C.3	Implementation approach for IDL	250
C.3.1	ASIS API server	250
C.3.2	ASIS API client tool	251
C.3.3	Approach to implement the Traverse_Element generic	251
C.3.4	IDL to implement the Traverse_Element generic	252
C.3.5	Ada code output by the IDL compiler	253

D RATIONALE	254
D.1 Benefits of code analysis	254
D.1.1 Definition	254
D.1.2 Applicability	254
D.1.3 Motivation	255
D.2 Technology for code analysis	255
D.2.1 Code parsers	255
D.2.2 DIANA	256
D.2.3 LRM-interface	257
D.2.4 ASIS	257
D.2.5 Benefits of ASIS standard	259
D.3 Design considerations for ASIS	259
D.3.1 Design goals	259
D.3.2 Major changes from ASIS for ISO 8652:1987	261
D.3.3 Essence of Ada and ASIS	261
D.4 Major issues regarding ASIS	262
D.4.1 Ada environment and compilation units	262
D.4.2 ASIS context and inconsistency	262
D.4.3 Implicit declarations	264
D.4.4 Abstract "=" for private types	264
D.4.5 Usage names and expressions	265
D.4.6 Select alternative	265
D.4.7 Attribute definition clauses	265
D.4.8 Configuration pragmas	265
D.4.9 Queries with additional context parameter	266
D.4.10 Ids	266
D.4.11 Data decomposition	266
D.5 Conclusion	267
D.6 Acronyms	268
 BIBLIOGRAPHY	 269
INDEX.....	270

Figures

Figure 1	ASIS implementation permissions	7
Figure 2	ASIS as interface to Ada compilation environment	10
Figure 3	Application interface to ASIS Context	11
Figure 4	Syntactic tree representation of an Ada object declaration	13
Figure 5	Operations on elements	15
Figure 6	Semantic reference using corresponding queries	17
Figure 7	ASIS package architecture	18
Figure C.1	Generation of client/server ASIS artifacts	250

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 reformatters, 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 reformatters, 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.

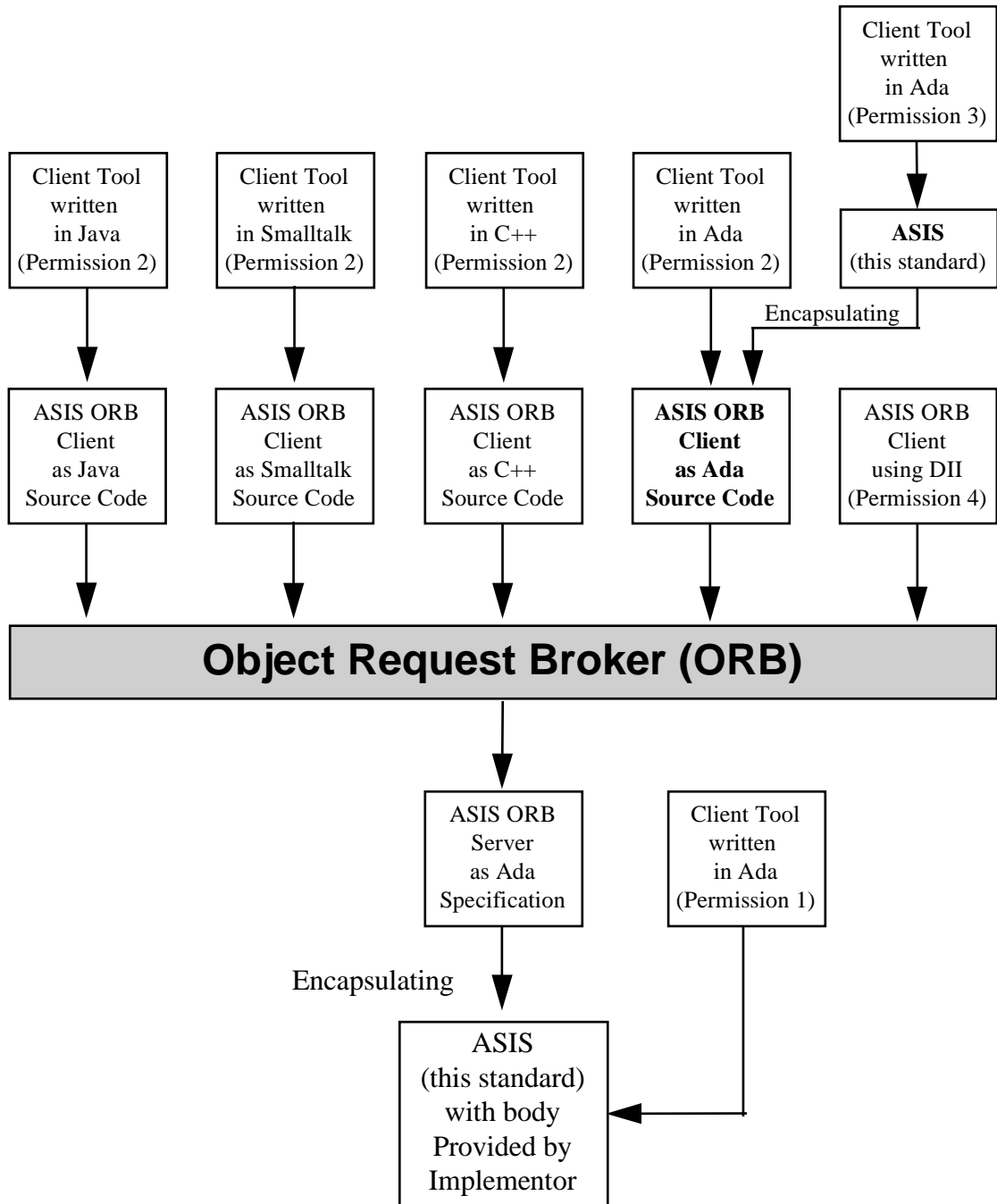


Figure 1 — ASIS implementation permissions

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:

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