
**Information technology — Programming
languages, their environments and
system software interfaces —
Programming language ISLISP**

*Technologies de l'information — Langages de programmation, leurs
environnements et interfaces de logiciel système — Langage de
programmation ISLISP*

iTeh STANDARD PREVIEW
(standards.iteh.ai)

ISO/IEC 13816:2007

<https://standards.iteh.ai/catalog/standards/sist/1adffd95-7b73-466f-a234-59a0c9190100/iso-iec-13816-2007>

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[ISO/IEC 13816:2007](https://standards.iteh.ai/catalog/standards/sist/1adffd95-7b73-466f-a234-59a0c9190100/iso-iec-13816-2007)

<https://standards.iteh.ai/catalog/standards/sist/1adffd95-7b73-466f-a234-59a0c9190100/iso-iec-13816-2007>



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2007

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 either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

1	Scope	1
2	Normative references	1
3	Compliance of ISLISP processors and text	2
4	Terms and definitions	2
5	Notation and conventions	7
6	Lexemes	9
6.1	Separators	10
6.2	Comments	10
7	Textual representation	10
8	Reserved identifiers	11
9	Errors	11
9.1	Classes of error specification	12
9.2	Pervasive error types	12
10	Classes	13
10.1	Metaclasses	13
10.2	Predefined classes	15
10.3	Standard classes	16
10.3.1	Slots	16
10.3.2	Creating instances of classes	16
11	Scope and extent	17
11.1	The lexical principle	17
11.2	Scope of identifiers	17
11.3	Some specific scope rules	18
11.4	Extent	18
12	Forms and evaluation	19
12.1	Forms	19
12.2	Function application forms	20
12.3	Special forms	20
12.4	Defining forms	21
12.5	Macro forms	21
12.6	The evaluation model	21
12.7	Functions	22
12.8	Defining operators	26
13	Predicates	28
13.1	Boolean values	28
13.2	Class predicates	28
13.3	Equality	29
13.4	Logical connectives	31
14	Control structure	33
14.1	Constants	33
14.2	Variables	34

ISO/IEC 13816:2007(E)

14.3	Dynamic variables	37
14.4	Conditional expressions	38
14.5	Sequencing forms	40
14.6	Iteration	41
14.7	Non-local exits	42
14.7.1	Establishing and invoking non-local exits	42
14.7.2	Assuring data consistency during non-local exits	46
15	Objects	47
15.1	Defining classes	47
15.1.1	Determining the class precedence list	50
15.1.2	Accessing slots	51
15.1.3	Inheritance of slots and slot options	51
15.2	Generic functions	52
15.2.1	Defining generic functions	52
15.2.2	Defining methods for generic functions	54
15.2.2.1	Agreement on parameter specializers and qualifiers	55
15.2.2.2	Congruent lambda-lists for all methods of a generic function	55
15.2.3	Inheritance of methods	55
15.3	Calling generic functions	56
15.3.1	Selecting the applicable methods	56
15.3.2	Sorting the applicable methods	56
15.3.3	Applying methods	57
15.3.3.1	Simple method combination	57
15.3.3.2	Standard method combination	57
15.3.4	Calling more general methods	59
15.4	Object creation and initialization	60
15.4.1	Initialize-object	60
15.5	Class enquiry	61
16	Macros	61
17	Declarations and coercions	63
18	Symbol class	65
18.1	Symbol names	65
18.1.1	Notation for symbols	65
18.1.2	Alphabetic case in symbol names	66
18.1.3	nil and ()	67
18.2	Symbol properties	67
18.3	Unnamed symbols	68
19	Number class	68
19.1	Number class	69
19.2	Float class	77
19.3	Integer class	80
20	Character class	83
21	List class	84
21.1	Cons	84
21.2	Null class	87
21.3	List operations	87
22	Arrays	91

22.1 Array classes	91
22.2 General arrays	92
22.3 Array operations	93
23 Vectors	95
24 String class	96
25 Sequence functions	99
26 Stream class	102
26.1 Streams to files	103
26.2 Other streams	105
27 Input and output	107
27.1 Argument conventions for input functions	107
27.2 Character I/O	107
27.3 Binary I/O	111
28 Files	112
29 Condition system	114
29.1 Conditions	115
29.2 Signaling and handling conditions	115
29.2.1 Operations relating to condition signaling	116
29.2.2 Operations relating to condition handling	117
29.3 Data associated with condition classes	118
29.3.1 Arithmetic errors	118
29.3.2 Domain errors	118
29.3.3 Parse errors	118
29.3.4 Simple errors	119
29.3.5 Stream errors	119
29.3.6 Undefined entity errors	119
29.4 Error identification	120
30 Miscellaneous	122
Index	123

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 13816 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

This second edition cancels and replaces the first edition (ISO/IEC 13816:1997), which has been technically revised.

Introduction

The programming language ISLISP is a member of the LISP family.

The following factors influenced the establishment of design goals for ISLISP:

1. A desire of the international LISP community to standardize on those features of LISP upon which there is widespread agreement.
2. The existence of the incompatible dialects COMMON-LISP, EULISP, LE-LISP, and SCHEME (mentioned in alphabetical order).
3. A desire to affirm LISP as an industrial language.

This led to the following design goals for ISLISP:

1. ISLISP shall be compatible with existing LISP dialects where feasible.
2. ISLISP shall have as a primary goal to provide basic functionality.
3. ISLISP shall be object-oriented. [ISO/IEC 13816:2007](https://standards.iteh.ai/catalog/standards/sist/1adfd95-7b73-466f-a234-100000000000/iso-iec-13816-2007)
4. ISLISP shall be designed with extensibility in mind. <https://standards.iteh.ai/catalog/standards/sist/1adfd95-7b73-466f-a234-100000000000/iso-iec-13816-2007>
5. ISLISP shall give priority to industrial needs over academic needs.
6. ISLISP shall promote efficient implementations and applications.

iTeh STANDARD PREVIEW **(standards.iteh.ai)**

ISO/IEC 13816:2007

<https://standards.iteh.ai/catalog/standards/sist/1adffd95-7b73-466f-a234-59a0c9190100/iso-iec-13816-2007>

Information technology — Programming languages, their environments and system software interfaces — Programming language ISLISP

1 Scope

This International Standard specifies syntax and semantics of the computer programming language ISLISP by specifying requirements for a conforming ISLISP processor and a conforming ISLISP text.

This International Standard does not specify:

- (a) the size or complexity of an ISLISP text that exceeds the capacity of any specific data processing system or the capacity of a particular processor, nor the actions to be taken when the corresponding limits are exceeded;
- (b) the minimal requirements of a data processing system that is capable of supporting an implementation of a processor for ISLISP;
- (c) the method of preparation of an ISLISP text for execution and the method of activation of this ISLISP text, prepared for execution;
- (d) the typographical presentation of an ISLISP text published for human reading;
- (e) extensions that might or might not be provided by the implementation.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- ISO/IEC TR 10034:1990, *Guidelines for the preparation of conformity clauses in programming language standards*
- IEEE standard 754-1985. *Standard for binary floating point arithmetic*

3 Compliance of ISLISP processors and text

An ISLISP processor complying with the requirements of this International Standard shall:

- (a) accept and implement all features of ISLISP specified in this International Standard;
- (b) reject any text that contains any textual usage which this International Standard explicitly defines to be a violation (see §9);
- (c) be accompanied by a document that provides the definitions of all implementation-defined features;
- (d) be accompanied by a document that separately describes any features accepted by the processor that are not specified in this International Standard; these extensions shall be described as being “extensions to ISLISP as specified by ISO/IEC 13816:2007(E).”

A complying ISLISP text shall not rely on implementation-dependent features. However, a complying ISLISP text may rely on implementation-defined features required by this International Standard.

A complying ISLISP text shall not attempt to create a lexical variable binding for any named constant defined in this International Standard. It is a violation if any such attempt is made.

iTeh STANDARD PREVIEW

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.
<https://standards.iteh.ai/catalog/standards/sist/1ad1dd95-7b73-460f-a234-59a0c9190100/iso-iec-13816-2007>

4.1

abstract class

class that by definition has no direct instances

4.2

activation

computation of a function

Note: Every activation has an activation point, an activation period, and an activation end. The activator, which is a function application form prepared for execution, starts the activation at the activation point.

4.3

accessor

association of a reader and a writer for a slot of an instance

4.4

argument position

occurrence of a text unit as an element in a form excluding the first one

4.5***binding***

concept that has both a syntactic and a semantic aspect, where

- syntactically, “binding” describes the relation between an identifier and a binding ISLISP form, and
- semantically, “binding” describes the relation between a variable, its denoting identifier, and an object (or, the relation between a variable and a location)

Note 1: The property of being bound can be checked textually by relating defining and applied identifier occurrences.

Note 2: Semantically, the binding relation might be imagined to be materialized in some entity, the binding. Such a binding entity is constructed at run time and destroyed later, or might have indefinite extent.

4.6***class***

object that determines the structure and behavior of a set of other objects called its instances

Note: The behavior is the set of operations that can be performed on an instance.

4.7***condition***

object that represents a situation that has been (or might be) detected by a running program

4.8***definition point***

textual point of an ISLISP text that is the start of an identifier's representation of an ISLISP object

4.9***direct instance***

instance of a class but not an instance of one of its subclasses

Note: Every ISLISP object is direct instance of exactly one class, which is called “its class”. The set of all direct instances together with their behavior constitute a class.

4.10***dynamic***

having an effect that is determined only through program execution and that cannot, in general, be determined statically

4.11***dynamic variable***

variable whose associated binding is determined by the most recently executed active block that established it, rather than statically by a lexically apparent block according to the lexical principle

4.12***evaluation***

computation of a form prepared for execution which results in a value and/or a side-effect

ISO/IEC 13816:2007(E)

4.13

execution

sequence of (sometimes nested) activations

4.14

extension

implementation-defined modification to the requirements of this International Standard that does not invalidate any ISLISP text complying with this International Standard (except by prohibiting the use of one or more particular spellings of identifiers), does not alter the set of actions which are required to signal errors, and does not alter the status of any feature designated as implementation dependent

4.15

form

single, syntactically valid unit of program text, capable of being prepared for execution

4.16

function

ISLISP object that is called with arguments, performs a computation (possibly having side-effects), and returns a value

4.17

generic function

function whose application behavior is determined by the classes of the values of its arguments and which consists in general of several methods

4.18

identifier

lexical element (lexeme) which designates an ISLISP object

Note: In the data structure representation of ISLISP texts, identifiers are denoted by symbols.

4.19

immutable binding

binding in which the relation between an identifier and the object represented by this identifier cannot be changed

Note: It is a violation if there is attempt to change an immutable binding (error-id. *immutable-binding*).

4.20

immutable object

object which is not subject to change, either because no operator is provided that is capable of effecting such change, or because some constraint exists which prohibits the use of an operator that might otherwise be capable of effecting such a change

Note: Except as explicitly indicated otherwise, a conforming processor is not required to detect attempts to modify immutable objects; the consequences are undefined if an attempt is made to modify an immutable object.

4.21

implementation defined

feature, possibly differing between different ISLISP processors, but completely defined for every processor

4.22***implementation dependent***

feature, possibly differing between different ISLISP processors, but not necessarily defined for any particular processor

Note: A conforming ISLISP text must not depend upon implementation-dependent features.

4.23***inheritance***

relation between a class and its superclass which maps structure and behavior of the superclass onto the class

Note: ISLISP supports a restricted form of multiple inheritance; *i.e.*, a class may have several direct superclasses at once.

4.24***instance***

(class) either a direct instance of a class or an instance of one of its subclasses

4.25***literal***

object whose representation occurs directly in a program as a constant value

4.26***metaclass***

class whose instances are themselves classes

4.27***method***

case of a generic function for a particular parameter profile, which defines the class-specific behavior and operations of the generic function

4.28***object***

anything that can be created, destroyed, manipulated, compared, stored, input, or output by the ISLISP processor

Note 1: In particular, functions are ISLISP objects.

Note 2: Objects that can be passed as arguments to functions, can be returned as values, can be bound to variables, and can be part of structures, are called *first-class objects*.

4.29***operator***

first element of a compound form, which is either a reserved name that identifies the form as a special form, or the name of a macro, or a lambda expression, or else an identifier in the function namespace

4.30***operator position***

occurrence of a text unit as the first element in a form

4.31

parameter profile

parameter list of a method, where each formal parameter is accompanied by its class name

Note: If a parameter is not accompanied by a class name, it belongs to the most general class.

4.32

place

location where objects can be stored and retrieved later

Note: Places are designated by forms which are permitted as the first argument of **setf**. If used this way an object is stored in the place. If the form is not used as first argument of **setf** the stored object is retrieved. The cases are listed in the description of **setf**.

4.33

process

execution of an ISLISP text prepared for execution

4.34

processor

system or mechanism, that accepts an ISLISP text (or an equivalent data structure) as input, prepares it for execution, and executes the result to produce values and side-effects

4.35

program

aggregation of expressions to be evaluated, the specific nature of which depends on context

Note: Within this International Standard, the term “program” is used only in an abstract way; there is no specific syntactic construct that delineates a program.

4.36

scope

(*identifier*) the textual part of a program where the meaning of that identifier is defined; *i.e.*, there exists an ISLISP object designated by this identifier

4.37

slot

named component of an instance which can be accessed using the slot accessors

Note: The structure of an instance is defined by the set of its slots.

4.38

text

text that complies with the requirements of this International Standard (*i.e.*, with the syntax and static semantics of ISLISP)

Note: An ISLISP text consists of a sequence of toplevel forms.

4.39

toplevel form

any form that either is not nested in any other form or is nested only in **progn** forms

4.40

toplevel scope

scope in which a complete ISLISP text unit is processed

4.41

writer

method associated with a slot of a class, whose task is to bind a value with a slot of an instance of that class

5 Notation and conventions

For a clear definition of, and a distinction between, syntactic and semantic concepts, several levels of description abstraction are used in the following.

There is a correspondence from ISLISP textual units to their ISLISP data structure representations. Throughout this International Standard the text and the corresponding ISLISP objects (data structures) are addressed simultaneously. ISLISP text can be seen as an external specification of ISLISP data structures. To distinguish between the two representations different concepts are used. When textual representation is discussed, textual elements (such as *identifiers*, *literals*, and *compound forms*) are used; when ISLISP objects are discussed, *objects* (such as *symbols* and *lists*) are used.

The constituents of ISLISP text are called **forms**. A **form** can be an *identifier*, a *literal*, or a *compound form*. A *compound form* can be a *function application form*, a *macro form*, a *special form*, or a *defining form*.

An *identifier* is represented by a *symbol*. A *compound form* is represented by a non-null *list*. A *literal* is represented by neither a symbol nor a list, and so is neither an *identifier* nor a *compound form*; for example, a number is a literal.

An object is **prepared for execution**; this might include transformation or compilation, including macro expansion. The method of preparation for execution and its result are not defined in this International Standard (with exception of the violations to be detected). After successful preparation for execution the result is ready for **execution**. The combination of preparation for execution and subsequent execution implements ISLISP's **evaluation model**. The term "evaluation" is used because ISLISP is an expression language—each form has a value which is used to compute the value of the containing form. The results obtained when an entity is prepared for execution are designated throughout this International Standard by the construction "prepared entity"; *e.g.*, "prepared form," "prepared special form."

Example: A "cond special form" becomes a "prepared cond" by preparation for execution.

In the examples, the metasymbol " \Rightarrow " designates the result of an actual evaluation. For example:

$$(+ \ 3 \ 4) \Rightarrow 7$$

The metasymbol " \rightarrow " identifies the class that results from the evaluation of a form having a given pattern. For example:

$$(+ \ i_1 \ i_2) \rightarrow \langle integer \rangle$$

Given a form pattern (usually defined by its constant parts, the function name or special operator), \rightarrow relates it to the class to which the result of the evaluation of all matching forms belong.