INTERNATIONAL STANDARD

ISO/IEC 13719-2

First edition
1995-06-01

Information technology — Portable Common Tool Environment (PCTE) —

Part 2:
C programming language binding

Technologies de l'information — Environnement d'outil courant portable (PCTE) —

Partie 2: Liant de langage de programmation C

Reference number
ISO/IEC 13719-2:1995(E)

Contents

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 13719-2 was prepared by the European Computer Manufacturers Association (ECMA) (as Standard ECMA-158) and was adopted, under a special "fast-track procedure", by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

ISO/IEC 13719 consists of the following parts, under the general title *Information technology — Portable Common Tool Environment (PCTE)*:

— *Part 1: Abstract specification*

— *Part 2: C programming language binding*

— *Part 3: Ada programming language binding*

Information technology — Portable Common Tool Environment (PCTE) —

Part 2:
C programming language binding

1 Scope

This part of ISO/IEC 13719 defines the standard binding of the Portable Common Tool Environment (PCTE), as specified in ISO/IEC 13719-1, to the C Programming Language.

A number of features are not completely defined in ISO/IEC 13719-1, some freedom being allowed to the implementor. Some of these features are specified as implementation limits. Some constraints are placed on these implementation limits by this C Language Binding Standard. These constraints are specified in clause 24, Implementation Limits.

PCTE is an interface to a set of facilities that forms the basis for constructing environments supporting systems engineering projects. These facilities are designed particularly to provide an infrastructure for programs which may be part of such environments. Such programs, which are used as aids to system development, are often referred to as tools.

2 Conformance

An implementation of PCTE conforms to this part of ISO/IEC 13719 if it conforms to 2.2 of ISO/IEC 13719-1, where the binding referred to there is taken to be the C Language Binding defined in clauses 1 to 5 and 8 to 25 of this part of ISO/IEC 13719. All other clauses in this part of ISO/IEC 13719 are provided as assistance to the reader and are not normative.

The C Language Binding defined in this part of ISO/IEC 13719 conforms to 2.1 of ISO/IEC 13719-1.

3 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 13719. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 13719 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 8601:1988,	*Data elements and interchange formats — Information interchange — Representation of dates and times.*
ISO 8859-1:1987,	*Information processing — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1.*
ISO 9899:1990,	*Programming languages — C.*
ISO/IEC TR 10182:1993,	*Information technology — Programming languages , their environments and system software interfaces — Guidelines for language bindings.*
ISO/IEC 11404:—[1],	*Information technology — Programming languages their environments and system software interfaces — Language-independent datatypes.*
ISO/IEC 13719-1:1995,	*Information technology — Portable Common Tool Environment (PCTE) — Part 1: Abstract specification.*

4 Definitions

Definitions used in this part of ISO/IEC 13719 are those defined in ISO/IEC 13719-1.

5 Formal notations

Two formal notations are used in this part of ISO/IEC 13719. For each bound operation, the abstract specification is given using a subset of the Vienna Development Method Specification Language (or VDM-SL), exactly as it is specified in ISO/IEC 13719-1. For the C Language binding for each operation, the function prototype syntax is used as defined in ISO 9899.

6 Outline of the standard

Clause 7 describes the strategy used to develop this binding specification.

Clause 8 contains the mapping from the datatypes that are used in the Abstract Specification to C Language datatypes.

Clauses 9 to 22 define the binding of datatypes and operations in the corresponding clauses of ISO/IEC 13719-1.

Clause 23 defines the binding of object, attribute, link, and type references, as specified in 23.1.2 and 23.2 of ISO/IEC 13719-1.

Clause 24 defines the binding of the implementation limit functions described in ISO/IEC 13719-1, clause 24.

Clause 25 defines the binding of the error conditions specified in ISO/IEC 13719-1, appendix C, and defines binding-defined error conditions for the C Binding.

---

1) To be published

7 Binding strategy

7.1 C Programming Language Standard

This part of ISO/IEC 13719 is a conforming program according to ISO 9899.

7.2 General Principles

The following general principles were applied when generating the binding in this part of ISO/IEC 13719.

ISO/IEC TR 10182 should be followed as far as possible for binding method 1: provide a completely defined procedural interface.

Each operation in ISO/IEC 13719-1 should be represented by exactly one operation in this part of ISO/IEC 13719 except possibly when the abstract operation has distinct functionality depending on the values of one or more parameters.

Each operation in this part of ISO/IEC 13719 should have the same number of parameters as does the corresponding operation in ISO/IEC 13719-1.

All operations in this part of ISO/IEC 13719 should return an integer status value. All other values returned by the operation should be passed back to the caller via an output parameter. The return value of the operation should indicate success or failure only.

Operation and parameter names should be the same in this part of ISO/IEC 13719 as they are in ISO/IEC 13719-1, with the exception that identifiers with file scope should begin 'Pcte_' and otherwise consist of lowercase letters and underscores. The PCTE standard guarantees that there are no ambiguities in names prefixed by 'Pcte_'.

All additional names introduced in this part of ISO/IEC 13719 which are visible across the interface (except header names, see 8.5) should begin 'Pcte_' and otherwise consist of lowercase letters, underscores and digits, or begin 'PCTE_' and otherwise consist of uppercase letters, underscores and digits. The PCTE standard guarantees that there are no ambiguities in names prefixed by 'PCTE_' or 'Pcte_'.

Wherever practical, types introduced for passing complex data entities between caller and operation (and vice versa) should be private types defined by this part of ISO/IEC 13719. The principle should only be ignored for reasons of ease of use and efficiency of implementation.

Each simple datatype in ISO/IEC 13719-1 should be mapped to a corresponding type defined in this part of ISO/IEC 13719. Each implementation of the binding should then be free to map the binding-defined type to an efficient C Language basic type appropriate for the platform of the implementation, within the constraints specified in this part of ISO/IEC 13719.

A general policy of memory allocation should be adopted; see 7.5

7.3 Sets and Sequences

Some Complex data entities to be passed into or retrieved from an operation are defined as sets or sequences of a base type in ISO/IEC 13719-1. Bounded set types are mapped individually to bit-significant natural numbers; unbounded set and sequence types are

mapped to a private type, **Pcte_sequence** with operations for creation, population, retrieval and deletion. These operations allow multiple elements of sets and sequences to be set and read in a single operation, from or to an array object of an appropriate base type. Thus, the data for sets and sequences can be easily manipulated using standard C Language paradigms, while allowing the implementation to choose the best implementation for such sets and sequences.

7.4 Character Strings

In ISO/IEC 13719-1, two different types are used to represent sequences of characters. String is a sequence of Octets allowing all 8-bit values and Text is a sequence of latin 1 graphic characters. Contents, string attributes etc., are of type String; keys, type names etc., are of type Text.

In the C Bindings, String is mapped to **Pcte_string** (see 8.2.11). Text is mapped to the native C language string with a possibly fixed length, i.e. char * with operations depending on NUL ('\0') character termination (see 8.2.6).

7.5 Memory Allocation

Communication between caller and operation is effected by the transfer of data into an operation via an input parameter or back from an operation via an output parameter. There are two types of such parameters: public and private.

All instances of a public type are allocated and managed by the caller of the operation. All instances of a private type are allocated and managed by the implementation. However, the extraction of data from a private type is again by data transfer in instances of a public type via an operation on the private type. In these cases also, the caller of the operation is required to allocate and manage the instances of the public type. The caller is further required to allocate sufficient space to contain a handle to the private type, the type of which is always a pointer to an internal data structure of undefined form. Operations on the private type are provided to create and discard these internal data structures.

Data stored in an instance of a private type is owned by the implementation. The implementation is responsible for allocating, managing and deallocating the memory used to store this data. Furthermore, after a handle to an instance of a private type has been returned to the caller of an operation, via an output parameter, the implementation is responsible for maintaining the data stored therein, until the caller explicitly indicates that the data is no longer needed, by invoking the discard operation on that instance.

Data stored in an instance of a public type and passed into an operation via an input parameter, is owned by the caller of the operation. The caller is responsible for allocating, managing and deallocating the memory used to store this data. The caller is further responsible for maintaining the data stored therein for the duration of the operation. If the implementation needs to access the data after the operation has completed, the implementation is responsible for allocating additional memory and storing therein a copy of the data.

All the operations that have sequences as "out" parameters will allocate the sequence and return it as result of operation. The user does not need to allocate the sequence in

advance: the user only needs to declare a Pcte_sequence variable and pass the address of that variable.

7.6 References and Names

Objects, attributes, links, and types are referred to in this part of ISO/IEC 13719 using object references, attribute references, link references and type references, respectively. References are mapped to private datatypes encapsulating two ways of designating an object, attribute, link, or type: by an external and by an internal reference (see ISO/IEC 13719-1, clause 23).

Beside these references, in this part of ISO/IEC 13719 also attribute names, link names and type names are used to refer to attributes, links, or types. These names represent external references and they are mapped to the native C language string type.

Therefore two different interfaces are provided in this Binding for clauses 9 to 22:

— one interface using names for attributes, links, and types.

— one interface using references for attributes, links, and types. All operations of this interface begin 'Pcte_h_'. These operations are defined if an operation of the previous interface uses attributes, links, or types. Whenever new datatypes are necessary, they also begin 'Pcte_h_' (see 8.7).

7.7 Operation Return Values

All the operations are mapped to functions which return a Pcte_error_type value, which indicates success (PCTE_NO_ERROR equivalent to PCTE_OK) or failure (one of the other enumeration values of Pcte_error_type) of the operation. All other information that is passed between the caller and the operation is passed via "out" or "in-out" parameters.

The error code PCTE_NO_ERROR is equivalent to PCTE_OK (as both have the value zero).

Additionally the global variable Pcte_error_number is set to the same return value of the function after each PCTE call, but it should be noted that in case of multi-threading or queue handlers this global variable cannot relied upon.

7.8 Error Conditions

Error conditions which are defined in ISO/IEC 13719-1 and which can be established and returned by the operations defined in this part of ISO/IEC 13719 are described in clause 25.

All binding defined errors are defined in 25.1(2).

7.9 Identifiers

Many of the identifiers in ISO/IEC 13719-1 are longer than 31 characters, but no two identifiers are exactly the same within the first 31 characters. The C Programming Language Standard requires that an internal name (i.e., a macro name or an identifier that does not have external linkage) be unique within the first 31 characters. Thus there is no need for any identifiers to be abbreviated in this Binding.

ISO 9899 requires all identifiers of enumeration values to be distinct. Where ISO/IEC 13719-1 uses the same identifier for values of enumeration types bound to different C Language enumeration types, new names have been invented.

In a few cases an abstract operation has been bound to more than one C Language function, to cater for optional parameters; in these cases also, new names have been invented.

7.10 Implementation Limits.

ISO/IEC 13719-1 defines a set of limits that must be honored by all implementations of the Language Bindings. Clause 24 describes the binding-defined identifiers for these limit values and the way in which these limits can be retrieved.

8 Datatype mapping

This clause defines the mapping of the parameter and result datatypes of the operations of ISO/IEC 13719-1 (*PCTE datatypes*) to the parameter and result datatypes of the operations of this part of ISO/IEC 13719 (*C datatypes*).

PCTE datatype names are printed in normal characters.

LI Datatypes names are printed in italics.

C datatype names are printed in bold except in displayed fragments of C.

The mapping from PCTE datatypes to C datatypes is done in two stages via LI datatypes defined in CD11404.2.

8.1 Mapping of PCTE Datatypes to LI Datatypes

As far as possible the names of PCTE datatypes are retained for the corresponding LI datatypes, but some new names are introduced.

The general strategy of this mapping is as follows.

- To select for each PCTE datatype a LI datatype definition which matches the requirements of the PCTE datatype defined in ISO/IEC 13719-1. The LI datatype definition is, where possible, a primitive LI datatype or otherwise a generated LI datatype.
- To define new datatype generators where needed.
- To map PCTE datatypes with the same properties to the same LI datatype.

8.1.1 Mapping of Predefined PCTE Datatypes

The mapping of these PCTE datatypes is as defined in ISO/IEC 13719-1, clause 23 and is summarized in table 1.

Table 1 - Mapping of predefined PCTE datatypes

PCTE datatype	LI datatype
Boolean	*boolean*
Integer	*pcte-integer = integer* range (*MIN_INTEGER_ATTRIBUTE*.. *MAX_INTEGER_ATTRIBUTE*)
Natural	*pcte-natural = integer* range (0..*MAX_NATURAL_ATTRIBUTE*)
Float	*pcte-float* = *real* (10, *MAX_DIGITS_FLOAT_ATTRIBUTE*) range (*MIN_FLOAT_ATTRIBUTE*.. *MAX_FLOAT_ATTRIBUTE*)
Time	*pcte-time* = *time* (*second*, 10, Pcte_accuracy_factor) range (*MIN_TIME_ATTRIBUTE*.. *MAX_TIME_ATTRIBUTE*)
Octet	*octet*
Text	*pcte-text = character string* (*repertoire*)
Enumerated type xxx = VALUE1|VALUE2|...	*pcte-xxx = enumerated*(value1, value2, ...)

8.1.2 Mapping of Private PCTE Datatypes

Table 2 - Mapping of private PCTE datatypes

PCTE datatype	LI datatype
Address	*address*
Attribute_reference	*attribute_reference*
Contents_handle	*contents_handle*
Handler	*handler*
Object_reference	*object_reference*
Link_reference	*link_reference*
Position_handle	*position_handle*
Profile_handle	*profile_handle*
Type_reference	*type_reference*

8.1.3 Mapping of Complex PCTE Datatypes

PCTE sequence datatypes are mapped to the new datatype generator *pcte-sequence* (see 8.1.4).

PCTE set datatypes are divided into bounded set types and unbounded set types. Bounded set types have values which are sets of enumeration values with at most 32 possible elements; all others are unbounded set types. Bounded set types are mapped via the new LI datatype generator *Bounded-set*. Unbounded set types are mapped via the new LI datatype generator Sequence, the order of elements being irrelevant.

When used as input parameter of an operation in clauses 9 to 22, a sequence which represents a PCTE unbounded set may contain repeated elements. The effect for the operation is as though each element occurred only once.

When returned as the result of an operation in clauses 9 to 22, an unbounded set has no repeated elements.

PCTE map are notionally mapped via a new LI datatype generator *Map*; their mappings to C datatypes are defined directly. The mapping of Attribute_assignments is defined in 9.1. The mapping of Access_rights, Acl, and Atomic_access_rights is defined in 19.1.

PCTE union datatypes other than enumerations are mapped via the datatype generator *Choice*.

PCTE composite and bracketed datatypes (except private PCTE datatypes, for which see 8.1.2) are mapped to the datatype generator *Record*.

8.1.4 New LI Datatype Generators

Pcte-Sequence

Description: *Pcte-sequence* is a datatype generator derived from *Sequence* by adding further characterizing operations. In some operations, an index of LI datatype *pcte-*

natural is used to identify elements in the sequence. The first element is always indexed from 0.

The characterizing operations are IsEmpty, Head, Tail, Equal, Empty, and Append from *Sequence* plus Get, Put, Copy, LengthOf, and IndexOf.

Get(s: sequence of *base*, index: Natural): *base* is undefined if InOrder(LengthOf(s), index) is true or is Head(s) if Equal(index, 0) is true; otherwise Get(Tail(s), Add(index, negate(1))).

Put(s: sequence of *base*, e: *base*, index: Natural): sequence of *base* is undefined if InOrder(Add(LengthOf(s), 1), index) is true or is Append(Create(e), s) if Equal(index, 0) is true; otherwise Append(Head(s), Put(Tail(s), e, Add(index, Negate(1)))).

Copy (s: sequence of *base*): sequence of *base* is Create() if IsEmpty(s) is true; otherwise Append(Head(s), Copy(Tail(s))).

LengthOf(s: sequence of *base*): Natural is 0 if IsEmpty(s); otherwise Add(LengthOf(Tail(s)), 1).

IndexOf(s: sequence of *base*, e: *base*): Natural is undefined if IsEmpty(s0 is true or is 1 if Equal(Head(s), e) is true; otherwise Add(IndexOf(Tail(s), e), 1).

Bounded-set

Description: Bounded-set is a datatype generator derived from *Set* by restricting the cardinality of the values to 32 or less.

bounded-set of (*base*) = new set of (*base*) : size (0..32);

The characterizing operations are IsIn, Subset, Equal, Difference, Union, Intersection, Empty, SetOf, Select from *Set*.