INTERNATIONAL STANDARD

ISO/IEC 13719-4

First edition
1998-10-01

Information technology — Portable Common Tool Environment (PCTE) —

Part 4:
IDL binding (Interface Definition Language)

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

Partie 4: Liaison IDL (langage de définition d'interface)

Reference number
ISO/IEC 13719-4:1998(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 13719-4 was prepared by ECMA (as Standard ECMA-230) 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*
- *Part 4: IDL binding (Interface Definition Language)*

Annex C forms an integral part of this part of ISO/IEC 13719. Annexes A and B are for information only.

Information technology - Portable Common Tool Environment (PCTE) -

Part 4:
IDL binding (Interface Definition Language)

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 CORBA Interface Definition Language (IDL) defined in ISO/IEC CD 14750.

A number of features are not completely defined in ISO/IEC 13719-1, some freedom being allowed to the implementer. Some of these features are specified as implementation limits. Some constraints are placed on these implementation limits by this IDL 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 systems 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 is taken to be the IDL 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 IDL 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/IEC 13719-1:1998, *Information technology - Portable Common Tool Environment (PCTE) - Part 1: Abstract specification.*

ISO/IEC 13719-2:1998, *Information technology - Portable Common Tool Environment (PCTE) - Part 2: C programming language binding.*

ISO/IEC 14750:—[1)], *Information technology - Open Distributed Processing - Interface Definition Language.*

1) To be published.

4 Definitions

All technical terms used in this part of ISO/IEC 13719, other than a few in widespread use, are defined in the body of this part of ISO/IEC 13719 or in the referenced documents.

5 Formal notations

For the IDL binding for each operation, the function syntax is used as defined in ISO/IEC CD 14750.

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 the IDL datatypes.

Clause 9 to 22 define the binding of datatypes and operations in the corresponding clauses of ISO/IEC 13719-1. The extensions for fine-grain objects are added at the end of clause 11.

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 clause 24 of ISO/IEC 13719-1.

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

There are 2 informative annexes. Annex A compares the structures of this IDL binding and of the C binding of ISO/IEC 13719-2, explaining the differences. Annex B describes the source file structure of the IDL binding.

Annex C, which is normative, contains the extensions for object orientation, corresponding to annex G of ISO/IEC 13719-1.

7 Binding strategy

7.1 IDL standard

This part of ISO/IEC 13719 conforms to the definition of IDL in ISO/IEC 14750.

7.2 General principles

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

The C interface generated from the IDL binding should be as close as possible to the PCTE C language binding of ISO/IEC 13719-2, so as to minimize changes to existing C applications.

The binding should leave open the possibility for an implementation of the binding to allow a non-PCTE process to access the PCTE object base without being statically linked to the PCTE

interface. This implies that the implementation of the static bindings generated from the IDL must not make use of any PCTE operations. The IDL binding has been structured, through the use of Pseudo-IDL (PIDL), to leave this implementation option open.

The majority of the operations accept a Pcte_object_reference as controlling object. Therefore, ideally, there should exist a Pcte_object_reference interface which inherits from almost all other interfaces. This approach would allow for a static type checking but it is awkward. It has been decided instead to allow casting and let the PCTE implementation raise an exception if the passed controlling object is not of the right type.

Many operations said to be applied to a process object are only applicable to the current process object. This is specified whenever it is necessary. This is also meant sometimes by the comment: /* Operation is applied to self */.

Sequences should be implemented as pseudo-objects to be mapped internally into CORBA sequences. This implies that each operation accepting or returning a sequence must map it in the correct format for the PCTE implementation server. In the case of a sequence of object or link references, each reference must be mapped to a CORBA interface and returned to the client as such. The major reason for this is that in general an object reference may not be easily mapped by an implementation into a format meaningful for network transport. It is easier to assume that the object references are kept on the implementation side, and that at the client side CORBA brings an object handle. This mapping allows the use of dynamic bindings as well as static bindings.

The possibility should be left open of a special implementation choice to implement the PCTE CORBA static bindings stubs to make direct use of the current PCTE C interface: this could be more efficient, but does not allow a distributed implementation of the IDL interface and might preclude the use of dynamic bindings.

7.3 Sets and sequences

All sequence operations are grouped under the Pcte_sequence interface. A difficulty is that the operation *create* is not part of the interface of an object. To keep the resulting generated C code in line with ISO/IEC 13719-2, it is still part of the Pcte_sequence interface, but the controlling object is a constant.

The input and/or result of a sequence create, insert, or get has been mapped to the IDL type **any**.

7.4 References and names

A departure from ISO/IEC 13719-1 is the introduction of an extra interface called PCTE_RF (Reference Factory), which contains those operations that return a reference but do not use a reference as a controlling object.

The rest of the mapping is straightforward, with three interfaces Pcte_object_reference, Pcte_link_reference, and Pcte_type_reference.

7.5 Implementation aspects

7.5.1 Source files

The source file structure is described in annex B. To simplify the IDL compilation process a few new IDL source files are introduced; this is because the ISO/IEC 13719-2 header structure includes both types and operations, where in many cases the latter are not needed. With IDL this leads to many forward references, eliminated by the introduction of oms_types.idl, discretionary_types.idl and mandatory_types.idl.

7.5.2 Naming changes in the IDL

All parameters with name containing 'attribute' have been renamed with 'attribute' replaced by 'attribute_ref'.

All parameters with name containing 'object' have been removed (i.e. as controlling object) or renamed with 'object' replaced by 'object_ref'.

The enumeration values PCTE_KEY, PCTE_NON_KEY to PCTE_KEY_ATTR, PCTE_NON_KEY_ATTR have been renamed to avoid clashes of the first item with Pcte_key, as IDL does not allow two identifiers which differ only by case to be used in the same scope.

The sequence enumeration items have been renamed to avoid clashes with the typedef of the sequences.

7.5.3 Difference in generated C code

All unions have extra '_d' and '_u' fields and are introduced by means of typedef. A result is that the resulting C code must be changed to use these extra fields.

The enumeration items cannot have a user-defined value. The generated header files must be changed manually.

8 Datatype mapping

8.1 Basic datatypes

The datatype mapping for basic types follows ISO/IEC 13719-2 closely.

- string is mapped to the IDL type **string**;
- natural and integer are mapped to the IDL type **long**;
- boolean is mapped to the IDL type **boolean**;
- float is mapped to the IDL type **float**;
- Pcte_pathname, Pcte_object_reference, etc. as identifier and interface name have been changed to be interfaces or pseudo-objects;

As IDL does not allow two identifiers which differ only by case to be used in the same scope, an "_EI" suffix has been added to the enumeration items of the Pcte_sequence_type (which otherwise would have been conflicting with sequence names).

8.2 Sequences

```
/* The source file "sequences.idl" */
#ifndef PCTE_SEQUENCES_INCLUDED
#define PCTE_SEQUENCES_INCLUDED 1
#include "types.idl"
enum Pcte_sequence_type {
    PCTE_ACCOUNTING_FILE_EI, PCTE_ACL_EI, PCTE_AUDIT_FILE_EI,
    PCTE_ATTRIBUTE_ASSIGNMENTS_EI, PCTE_H_ATTRIBUTE_ASSIGNMENTS_EI,
    PCTE_ATTRIBUTE_NAMES_EI, PCTE_ATTRIBUTE_REFERENCES_EI,
    PCTE_BUFFER_EI, PCTE_CONFIDENTIALITY_CRITERIA_EI,
    PCTE_ENUMERATION_VALUE_TYPE_EI,
    PCTE_H_ENUMERATION_VALUE_TYPE_EI,
    PCTE_ENUMERATION_VALUE_TYPE_IN_SDS_EI, PCTE_GENERAL_CRITERIA_EI,
    PCTE_INTEGRITY_CRITERIA_EI, PCTE_KEY_TYPES_EI, PCTE_H_KEY_TYPES_EI,
    PCTE_KEY_TYPES_IN_SDS_EI, PCTE_LINK_NAMES_EI,
    PCTE_LINK_SET_DESCRIPTORS_EI, PCTE_H_LINK_SET_DESCRIPTORS_EI,
    PCTE_LINK_REFERENCES_EI, PCTE_MESSAGE_TYPES_EI,
    PCTE_NAME_SEQUENCE_EI, PCTE_OBJECT_CRITERIA_EI,
    PCTE_OBJECT_REFERENCES_EI, PCTE_TYPE_NAMES_EI,
    PCTE_TYPE_NAMES_IN_SDS_EI, PCTE_TYPE_REFERENCES_EI,
    PCTE_USER_CRITERIA_EI, PCTE_VOLUME_INFOS_EI,

    /* New Object-Oriented extension sequences */
    PCTE_PARAMETER_ITEMS_EI,
    PCTE_METHOD_REQUESTS_EI,
    PCTE_CONTEXT_ADOPTIONS_EI,
    PCTE_METHOD_REQUEST_IDS_EI
    };
typedef Object Pcte_sequence_element;
typedef Object Pcte_array_of_sequence_elements;
interface Pcte_sequence;
#define Pcte_null_sequence (Pcte_sequence) NULL
typedef Pcte_sequence Pcte_accounting_file;
typedef Pcte_sequence Pcte_audit_file;
typedef Pcte_sequence Pcte_attribute_names;
typedef Pcte_sequence Pcte_attribute_references;
typedef Pcte_sequence Pcte_buffer;
typedef Pcte_sequence Pcte_confidentiality_criteria;
typedef Pcte_sequence Pcte_enumeration_value_type;
typedef Pcte_sequence Pcte_h_enumeration_value_type;
typedef Pcte_sequence Pcte_enumeration_value_type_in_sds;
```

```
typedef Pcte_sequence Pcte_general_criteria;
typedef Pcte_sequence Pcte_integrity_criteria;
typedef Pcte_sequence Pcte_key_types;
typedef Pcte_sequence Pcte_h_key_types;
typedef Pcte_sequence Pcte_key_types_in_sds;
typedef Pcte_sequence Pcte_link_set_descriptors;
typedef Pcte_sequence Pcte_h_link_set_descriptors;
typedef Pcte_sequence Pcte_link_names;
typedef Pcte_sequence Pcte_link_references;
typedef Pcte_sequence Pcte_message_types;
typedef Pcte_sequence Pcte_name_sequence;
typedef Pcte_sequence Pcte_object_criteria;
typedef Pcte_sequence Pcte_object_references;
typedef Pcte_sequence Pcte_type_names;
typedef Pcte_sequence Pcte_type_names_in_sds;
typedef Pcte_sequence Pcte_type_references;
typedef Pcte_sequence Pcte_user_criteria;
typedef Pcte_sequence Pcte_volume_infos;
typedef Pcte_sequence Pcte_parameters_items;
typedef Pcte_sequence Pcte_method_requests;
typedef Pcte_sequence Pcte_method_requests;
typedef Pcte_sequence Pcte_method_request_ids;
interface Pcte_sequence {                                          //PIDL
/* Mapped to a CORBA sequence. */
/* This interface is conventionally applied to the PCTE object type "process". */
Pcte_error_type create (
    in Pcte_sequence_type                  type,
    in Pcte_array_of_sequence_elements     data,
    in Pcte_natural                        count,
    out Pcte_sequence                      out_sequence
);
Pcte_error_type discard (
);
```

```
Pcte_error_type copy (
    out Pcte_sequence    destination_list,
    in Pcte_natural      index,
    in Pcte_natural      source_index,
    in Pcte_natural      count
);

Pcte_error_type insert_elements (
    in Pcte_natural                          index,
    in Pcte_array_of_sequence_elements       data,
    in Pcte_natural                          count
);

Pcte_error_type delete (
    in Pcte_natural    index,
    in Pcte_natural    count
);

Pcte_error_type are_equal (
    in Pcte_sequence second_sequence,
    out Pcte_boolean     equality
);

Pcte_error_type get_index (
    in Pcte_sequence_element   element,
    out Pcte_integer           index
);

Pcte_error_type get_length (
    out Pcte_natural  length
);

Pcte_error_type get_elements (
    in Pcte_natural                          index,
    out Pcte_array_of_sequence_elements      data,
    in Pcte_natural                          count
);

Pcte_error_type get (
    in Pcte_natural                index,
    out Pcte_sequence_element      element
);

Pcte_error_type insert (
    in Pcte_natural            index,
    in Pcte_sequence_element   element
);

Pcte_error_type replace (
    in Pcte_natural            index,
    in Pcte_sequence_element   element
);
```

```
Pcte_error_type append (
    in Pcte_sequence_element   element
);

Pcte_error_type normalize (
);

};

#endif
```

8.3 The global pcte source file

```
/* The source file "pcte.idl" */

#ifndef PCTE_INCLUDED
#define PCTE_INCLUDED 1

#include   "types.idl"          // 8.4
#include   "sequences.idl"      // 8.2
#include   "references.idl"     // clause 23
#include   "limits.idl"         // clause 24
#include   "errors.idl"         // clause 25

#include   "oms.idl"            // clause 9
#include   "sms.idl"            // clause 10
#include   "devices.idl"        // clause 11
#include   "contents.idl"       // clause 12
#include   "execution.idl"      // clause 13
#include   "messages.idl"       // clause 14
#include   "notification.idl"   // clause 15
#include   "activities.idl"     // clause 16
#include   "replication.idl"    // clause 17
#include   "network.idl"        // clause 18
#include   "discretionary.idl"  // clause 19
#include   "mandatory.idl"      // clause 20
#include   "auditing.idl"       // clause 21
#include   "accounting.idl"     // clause 22

/* #include directive used for cluster management */

#include   "clusters.idl"

/* #include directives used by Pcte object-oriented extensions */

#include   "interfaces.idl"
#include   "methods.idl"

 #endif      // ! PCTE_INCLUDED
```

8.4 The PCTE basic type source file

```
/* The source file "types.idl" */
#ifndef PCTE_TYPES_INCLUDED
#define PCTE_TYPES_INCLUDED 1
typedef unsigned long time_t;
#include "errors.idl"
#define PCTE_OK 0
#define PCTE_ERROR 1
typedef unsigned short Pcte_boolean;
#define PCTE_TRUE (Pcte_boolean) 1
#define PCTE_FALSE (Pcte_boolean) 0
typedef long            Pcte_integer;
typedef unsigned long   Pcte_natural;
typedef float           Pcte_float;
typedef time_t          Pcte_time;
#define Pcte_time_accuracy_factor (Pcte_natural) <implementation-defined>
#define Pcte_reference_time (Pcte_time) <implementation-defined>
#define Pcte_null_time (Pcte_time) <implementation-defined>
typedef octet Pcte_octet;
struct Pcte_string {
Pcte_natural   size;
Pcte_octet array;
};
#endif      // !PCTE_TYPES_INCLUDED
```


9 Object management

9.1 Object management datatypes

```
/* The source file "oms_types.idl" */
#define PCTE_OMS_TYPES_INCLUDED 1
enum Pcte_category {
    PCTE_COMPOSITION,
    PCTE_EXISTENCE,
    PCTE_REFERENCE,
    PCTE_DESIGNATION,
    PCTE_IMPLICIT
};
typedef Pcte_natural Pcte_categories ;
```