Standard ECMA-256
December 1996

ECMA

Standardizing Information and Communication Systems

Portable Common Tool Environment (PCTE) - Object-Orientation Extensions - C Programming Language Binding

Phone: +41 22 849.60.00 - **Fax:** +41 22 849.60.01 - **URL:** http://www.ecma.ch - **Internet:** helpdesk@ecma.ch

Standard ECMA-256
December 1996

ECMA

Standardizing Information and Communication Systems

Portable Common Tool Environment (PCTE) - Object-Orientation Extensions - C Programming Language Binding

Phone: +41 22 849.60.00 - **Fax:** +41 22 849.60.01 - **URL:** http://www.ecma.ch - **Internet:** helpdesk@ecma.ch

GL Ecma-256.doc 28-02-97 12,07

Brief History

With the development of object-oriented methods and programming languages, there is an increasing demand to enhance PCTE with the ability to represent active objects, i.e. objects characterised by interfaces defining the operations which are applicable to the objects of a given type, and define their dynamic behaviour.

In 1993, several projects addressed this problem. Two of them produced results which were made publicly available and were thereafter used as input to this Standard:

- the Portable Common Interface Set (PCIS) project of the NATO Special Working Group on APSE,
- the Object Oriented Tool Interface Set (OOTIS) project of IBM.

By the end of 1993, the US Department of Defense, the US National Institute of Standards and Technology, and the Object Management Group (OMG) decided to create an initiative, called the North American PCTE Initiative (NAPI) in order to resolve this problem (among others).

At the same time, the technical committee TC33 of ECMA decided to create a new working group, named TGOO, to add object orientation and support of fine-grain objects to PCTE. The NAPI and TGOO working groups soon decided to merge their efforts in order to do a joint specification.

In 1994, the NAPI group transformed itself into the OMG Special Interest Group on PCTE (OMG PCTE SIG) and the joint work with ECMA TC33/TGOO continued.

In September 1994, a new working group ISO/IEC JTC1/SC22/WG22 was created to manage the maintenance of the PCTE International Standard ISO/IEC 13719, which is equivalent to ECMA-149, 3rd edition. That working group participated to the review of the final drafts of this Standard.

This Standard is the result of all these collaborative efforts.

This Standard has been adopted by the ECMA General Assembly of December 1996.

Table of contents

1 Scope

(1) This document defines the standard binding of the Portable Common Tool Environment (PCTE) extensions for the support of object-orientation as specified in ECMA-255.

2 Conformance

(1) An implementation of PCTE conforms to this Standard if it conforms to both ECMA-158 and to ECMA-255, as defined in 2.2 of that Standard, where the binding referred is taken to be the C Binding defined in clauses 1 to 5 and 8 to 11 of this Standard. Parts 6 and 7 of this Standard are provided as assistance to the reader and are not normative.

(2) The C Binding defined in this Standard conforms to ECMA-255, as defined in 2.1 of that Standard.

3 Normative references

(1) The following standards contain provisions which, through reference in this text, constitute provisions of this Standard. At the time of publication, the editions indicated were 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 editions of the standards indicated below:

(2)	ISO/IEC 9899 : 1990	Information technology - Programming languages, their environments and system software interfaces - C programming language
(3)	ECMA-149	Portable Common Tool Environment (PCTE) - Abstract Specification (3rd Edition, December 1994)
(4)	ECMA-158	Portable Common Tool Environment (PCTE) - C Language Binding (3rd Edition, December 1994)
(5)	ECMA-255	Portable Common Tool Environment (PCTE) - Object-Orientation Extensions - Abstract Specification (1st edition, December 1996)

4 Definitions

(1) All technical terms used in this standard, other than a few in widespread use, are defined in the body of this standard or in ECMA-149, ECMA-158, ECMA-255, or ISO/IEC 9899.

5 Formal notations

(1) The notations used in this Standard are the same as those used in ECMA-158.

6 Outline of the standard

(1) Clause 6 gives an overview of the document and of the structure of the definition.

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

(3) Clause 8 contains the mapping of datatypes used in ECMA-255.

(4) Clauses 9 to 10 defines the binding of datatypes and operations in the corresponding clauses of ECMA-255.

(5) Clause 11 extends the binding of the error conditions specified in ECMA-158, clause 25.

7 Binding strategy

(1) The binding strategy used in this Standard is the same as is used in ECMA-158.

8 Datatype mapping

(1) The datatype mapping used in this Standard is the same as is used in ECMA-158, with some extensions.

(2) The sequence types defined in 8.5 of ECMA-158 are extended with four new sequences types:

(3)
```
typedef Pcte_sequence Pcte_parameters_items;
```

(4)
```
typedef Pcte_sequence Pcte_method_requests;
```

(5)
```
typedef Pcte_sequence Pcte_context_adoptions;
```

(6)
```
typedef Pcte_sequence Pcte_method_request_ids;
```

(7) The enumeration datatype **Pcte_sequence_type** defined in 8.5 of ECMA-158 is extended as follows:

(8)
```
typedef enum {
        PCTE_ACCOUNTING_FILE,
        ...
        PCTE_VOLUME_INFOS,

/* New Object-Oriented extension sequences */

        PCTE_PARAMETER_ITEMS,
        PCTE_METHOD_REQUESTS,
        PCTE_CONTEXT_ADOPTIONS,
        PCTE_METHOD_REQUEST_IDS
} Pcte_sequence_type;
```

(9) These additional values of **Pcte_sequence_type** are used to indicate the C element type of the sequence as follows:

enumeration value	C element type
PCTE_PARAMETER_ITEMS	Pcte_parameter_item
PCTE_METHOD_REQUESTS	Pcte_method_request
PCTE_CONTEXT_ADOPTIONS	Pcte_context_adoption
PCTE_METHOD_REQUEST_IDS	Pcte_method_request_id

(10) An implementation of this Standard must extend the global header file <Pcte/sequences.h> defined in 8.7.3 of ECMA-158 with the new declaration of **Pcte_sequence_type** defined above.

(11) An implementation of this Standard must extend the global header file <Pcte/Pcte.h> as follows:

(12)
```
/* The header <Pcte/pcte.h> */
```

(13)
```
#ifndef PCTE_INCLUDED
#define PCTE_INCLUDED 1
```

(14)
```
#include <Pcte/types.h>
```

(15)
```
/* All #include directives of 8.7.1 of ECMA-158 are included here. */

#include <Pcte/types.h>
#include <Pcte/sequences.h>
...
...
#include <Pcte/accounting.h>
```

(16)
```
/* #include directives used by Pcte object-oriented extensions */
```

(17)
```
#include <Pcte/interfaces.h>
```

(18)
```
#include <Pcte/methods.h>
```

(19)
```
#endif
```

9 Object-oriented invocation management

(1) /* The header <Pcte/methods.h> */

(2) #ifndef PCTE_IMPLEMENTATIONS_INCLUDED
#define PCTE_IMPLEMENTATIONS_INCLUDED 1

(3) #include <Pcte/types.h>

(4) #include <Pcte/references.h>

(5) #include <Pcte/sequences.h>

(6) #include <Pcte/oms.h>

9.1 Object-oriented invocation management datatypes

(1)
```
typedef enum {
        PCTE_CONSTRAINED_TO_ATTRIBUTE,
        PCTE_CONSTRAINED_TO_OBJECT,
        PCTE_CONSTRAINED_TO_INTERFACE
        } Pcte_parameter_constraint;
```

(2)
```
typedef struct {
        Pcte_parameter_constraint       constraint;
        union {
                Pcte_attribute_value    *p_value;
                Pcte_object_reference   p_object;
                Pcte_object_reference   p_interface;
        } parameter;
} Pcte_parameter_item;
```

(3) typedef Pcte_sequence Pcte_parameter_items;

(4)
```
typedef struct {
        Pcte_object_reference   target_object;
        Pcte_type_name          operation_id;
        Pcte_parameter_items    parameters;
        Pcte_object_reference   context;
} Pcte_method_request;
```

(5) typedef Pcte_sequence Pcte_method_requests;

(6)
```
typedef enum {
        PCTE_ADOPT_WORKING_SCHEMA       =       1<<0,
        PCTE_ADOPT_ACTIVITY             =       1<<1,
        PCTE_ADOPT_USER                 =       1<<2,
        PCTE_ADOPT_OPEN_OBJECTS         =       1<<3,
        PCTE_ADOPT_REFERENCE_OBJECTS    =       1<<4,
        PCTE_ADOPT_ALL
} Pcte_context_adoption;
```

(7)
```
#define PCTE_ADOPT_ALL (Pcte_natural)
        (PCTE_ADOPT_WORKING_SCHEMA | \
        PCTE_ADOPT_ACTIVITY | \
        PCTE_ADOPT_USER | \
        PCTE_ADOPT_OPEN_OBJECTS | \
        PCTE_ADOPT_REFERENCE_OBJECTS)
```

(8) typedef Pcte_sequence Pcte_context_adoptions;

(9) typedef void *Pcte_method_request_id;

(10) typedef Pcte_sequence Pcte_method_request_ids;

9.2 Object-oriented invocation management operations

```
/* 9.2.1 PROCESS_ADOPT_CONTEXT */

(1)  int Pcte_process_adopt_context (
       Pcte_context_adoptions          context_adoptions;
     )

     /* 9.2.2 REQUEST_INVOKE */

(2)  int Pcte_request_invoke (
       Pcte_method_request             *request,
       Pcte_context_adoptions          context_adoptions;
       Pcte_method_request_id          *request_id;
     )

     /* 9.2.3 REQUEST_SEND */

(3)  int Pcte_request_send (
       Pcte_method_request             *request,
       Pcte_context_adoptions          context_adoptions;
       Pcte_method_request_id          *request_id;
     )

     /* 9.2.4 REQUEST_SEND_MULTIPLE */

(4)  int Pcte_request_send_multiple (
       Pcte_method_requesta            requests,
       Pcte_context_adoptions          context_adoptions;
       Pcte_method_request_ids         *request_ids;
     )

(5)  #endif
```

10 Object-oriented schema management

```
(1)  /* The header <Pcte/interfaces.h> */

(2)  #ifndef PCTE_INTERFACES_INCLUDED
     #define PCTE_INTERFACES_INCLUDED 1

(3)  #include <Pcte/references.h>

(4)  #include <Pcte/sequences.h>
```

10.1 Object-oriented schema management datatypes

```
(1)  typedef enum {
           PCTE_NO_OPERATION, PCTE_ALL_OPERATIONS
     } Pcte_interface_scope;
```

10.2 Object-oriented schema management operations

```
     /* 10.2.1 SDS_APPLY_INTERFACE_TYPE */

(1)  int Pcte_sds_apply_interface_type (
       Pcte_object_reference           sds,
       Pcte_type_name_in_sds           interface_type,
       Pcte_type_name_in_sds           type
     );
```

```
/* 10.2.2 SDS_APPLY_OPERATION_TYPE */

(2)  int Pcte_sds_apply_operation_type (
        Pcte_object_reference          sds,
        Pcte_type_name_in_sds          operation_type,
        Pcte_type_name_in_sds          type
     );

     /* 10.2.3 SDS_CREATE_DATA_PARAMETER_TYPE */

(3)  int Pcte_sds_create_data_parameter_type (
        Pcte_object_reference          sds,
        Pcte_name                      local_name,
        Pcte_type_name                 data_type,
        Pcte_type_name                 new_parameter
     );

(4)  /*  The effect of not providing the optional parameter local_name to the*/
     /*  abstract operation is achieved by specifying local_name as NULL.  */

     /* 10.2.4 SDS_CREATE_INTERFACE_PARAMETER_TYPE */

(5)  int Pcte_sds_create_interface_parameter_type (
        Pcte_object_reference          sds,
        Pcte_name                      local_name,
        Pcte_type_name                 interface_type,
        Pcte_type_name                 new_parameter
     );

(6)  /*  The effect of not providing the optional parameter local_name to the*/
     /*  abstract operation is achieved by specifying local_name as NULL.  */

     /* 10.2.5 SDS_CREATE_INTERFACE_TYPE */

(7)  int Pcte_sds_create_interface_type (
        Pcte_object_reference          sds,
        Pcte_name                      local_name,
        Pcte_types_names_in_sds        parents,
        Pcte_types_names_in_sds        new_operations,
        Pcte_type_name_in_sds          new_interface
     );

(8)  /*  The effect of not providing the optional parameter local_name to the*/
     /*  abstract operation is achieved by specifying local_name as NULL.  */

     /* 10.2.6 SDS_CREATE_OBJECT_PARAMETER_TYPE */

(9)  int Pcte_sds_create_object_parameter_type (
        Pcte_object_reference          sds,
        Pcte_name                      local_name,
        Pcte_type_name                 object_type,
        Pcte_type_name                 new_parameter
     );

(10) /*  The effect of not providing the optional parameter local_name to the*/
     /*  abstract operation is achieved by specifying local_name as NULL.  */
```

```
/* 10.2.7 SDS_CREATE_OPERATION_TYPE */
(11) int Pcte_sds_create_operation_type (
       Pcte_object_reference          sds,
       Pcte_name                      local_name,
       Pcte_types_names_in_sds        parameters,
       Pcte_type_name_in_sds          return_value,
       Pcte_type_name_in_sds          new_operation
     );
(12) /* The effect of not providing the optional parameter local_name to the*/
     /* abstract operation is achieved by specifying local_name as NULL.  */
     /* 10.2.8 SDS_IMPORT_INTERFACE_TYPE */
(13) int Pcte_sds_import_interface_type (
       Pcte_object_reference          to_sds,
       Pcte_object_reference          from_sds,
       Pcte_type_name_in_sds          type,
       Pcte_name                      local_name,
       Pcte_interface_scope           import_scope
     );
(14) /* The effect of not providing the optional parameter local_name to the*/
     /* abstract operation is achieved by specifying local_name as NULL.  */
     /* 10.2.9 SDS_IMPORT_OPERATION_TYPE */
(15) int Pcte_sds_import_operation_type (
       Pcte_object_reference          to_sds,
       Pcte_object_reference          from_sds,
       Pcte_type_name_in_sds          type,
       Pcte_name                      local_name
     );
(16) /* The effect of not providing the optional parameter local_name to the*/
     /* abstract operation is achieved by specifying local_name as NULL.  */
     /* 10.2.10 SDS_UNAPPLY_INTERFACE_TYPE */
(17) int Pcte_sds_unapply_interface_type (
       Pcte_object_reference          sds,
       Pcte_type_name_in_sds          interface_type,
       Pcte_type_name_in_sds          type
     );
     /* 10.2.11 SDS_UNAPPLY_OPERATION_TYPE */
(18) int Pcte_sds_unapply_operation_type (
       Pcte_object_reference          sds,
       Pcte_type_name_in_sds          operation_type,
       Pcte_type_name_in_sds          type
     );
(19) #endif
```

11 Error conditions

(1) An implementation of this Standard must extend the header file <Pcte/errors.h> as follows:

```
(2) /* The header <Pcte/errors.h> */
(3) #ifndef PCTE_ERRORS_INCLUDED
    #define PCTE_ERRORS_INCLUDED 1
```

(4)
```
typedef enum {
        PCTE_NO_ERROR,

        /* All errors defined in 25.1 of ECMA-158 are here */

        PCTE_ACCESS_MODE_IS_INCOMPATIBLE,
        PCTE_ACCESS_MODE_IS_NOT_ALLOWED,
...
        PCTE_WORKSTATION_IS_UNKNOWN,

        /* New Object-oriented extension errors */

        PCTE_NUMBER_OF_PARAMETERS_IS_WRONG,
        PCTE_OPERATION_METHOD_CANNOT_FOUND,
        PCTE_OPERATION_METHOD_CANNOT_BE_ACTIVATED,
        PCTE_TYPE_IS_ALREADY_CONSTRAINED,
        PCTE_TYPE_OF_PARAMETER_IS_WRONG,

        /* C binding specific errors */

        PCTE_ACCESS_MASK_IS_INVALID,
...
        PCTE_VALUE_TYPE_IDENTIFIER_DOES_NOT_MATCH
} Pcte_error_type;
```

(5)
```
#endif
```

ECMA

114 Rue du Rhône
CH-1204 Geneva
Switzerland

This Standard ECMA-256 is available free of charge in printed form{ and as a file}.

See inside cover page for instructions