Standard ECMA-257

December 1996

ECMA

Standardizing Information and Communication Systems

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

Standard ECMA-257
December 1996

ECMA

Standardizing Information and Communication Systems

Portable Common Tool Environment (PCTE) - Object-Orientation Extensions - Ada 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-257.doc 28-02-97 12,10

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 characterized 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**

2 Conformance **1**

3 Normative references **1**

4 Definitions **1**

5 Formal notations **1**

6 Outline of the standard **1**

7 Binding strategy **1**

8 Datatype mapping **1**

9 Object-oriented invocation management **2**
9.1 Object-oriented invocation management datatypes 2
9.2 Object-oriented invocation management operations 7

10 Object-oriented schema management **8**
10.1 Object-oriented schema management datatypes 8
10.2 Object-oriented schema management operations 8

11 Error conditions **9**

1 Scope

(1) This ECMA Standard defines the standard binding of the Portable Common Tool Environment (PCTE) extensions for the support of object-orientation as specified in ECMA-255, to the Ada Programming Language. It forms an extension to ECMA-162.

2 Conformance

(1) An implementation of PCTE conforms to this Standard if it conforms to both ECMA-162 and to 2.1 of ECMA-255, where the binding referred is taken to be the Ada Binding defined in clause 1 to 5 and 8 to 11 of this Standard.

(2) The Ada Language Binding defined in this Standard conforms to 2.1 of ECMA-255.

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 8652 : 1995	Information technology - Programming languages, their environments and system software interfaces - Programming language Ada, (referring to ANSI-MIL-STD 1815A - Reference Manual for the Ada Programming Language)
(3)	ECMA-149	Portable Common Tool Environment (PCTE) - Abstract Specification (3rd edition, December 1994)
(4)	ECMA-162	Portable Common Tool Environment (PCTE) - Ada Programming 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-162, ECMA-255, or ISO/IEC 8652.

5 Formal notations

(1) The notations used in this Standard are the same as those used in ECMA-162.

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-162, clause 25.

7 Binding strategy

(1) The binding strategy used in this Standard is the same as that used in ECMA-162.

8 Datatype mapping

(1) The datatype mapping used in this Standard is the same as that used in ECMA-162.

9 Object-oriented invocation management

(1) **with** Pcte, Pcte_error;

(2) **package** Pcte_oo **is**

(3) **use** Pcte, Pcte_error;

9.1 Object-oriented invocation management datatypes

```
(1)     type parameter_constraint is (
                CONSTRAINED_TO_ATTRIBUTE,
                CONSTRAINED_TO_OBJECT,
                CONSTRAINED_TO_INTERFACE);

(2)     type parameter_item is record (
                constraint : Pcte_oo.parameter_constraint := CONSTRAINED_TO_ATTRIBUTE)
        is record
                case constraint is
                        when CONSTRAINED_TO_ATTRIBUTE =>
                                value     : Pcte.attribute_value;
                        when CONSTRAINED_TO_OBJECT =>
                                object    : Pcte.object_reference;
                        when CONSTRAINED_TO_INTERFACE =>
                                interface : Pcte.object_reference;
                end case;
        end record;

(3)     --      The semantics of the operations of this package are defined in 8.2.8 of ECMA-149.

(4)     package parameter_items is

(5)             type sequence is limited private;

(6)             --      Pcte_oo.parameter_items.sequence corresponds to the PCTE datatype
                --      Parameter_items.

(7)             function get (
                        list    : Pcte_oo.parameter_items.sequence;
                        index   : Pcte.natural := Pcte.natural'FIRST;
                        status  : Pcte_error.handle := EXCEPTION_ONLY)
                        return  Pcte.name;

(8)             procedure insert (
                        list    : in out Pcte_oo.parameter_items.sequence;
                        item    : in     Pcte_oo.parameter_item;
                        index   : in     Pcte.natural := Pcte.natural'LAST;
                        status  : in     Pcte_error.handle := EXCEPTION_ONLY);

(9)             procedure replace (
                        list    : in out Pcte_oo.parameter_items.sequence;
                        item    : in     Pcte_oo.parameter_item;
                        index   : in     Pcte.natural := Pcte.natural'LAST;
                        status  : in     Pcte_error.handle := EXCEPTION_ONLY);

(10)            procedure append (
                        list    : in out Pcte_oo.parameter_items.sequence;
                        item    : in     Pcte_oo.parameter_item;
                        status  : in     Pcte_error.handle := EXCEPTION_ONLY);
```

```
(11)        procedure delete (
                list    : in out  Pcte_oo.parameter_items.sequence;
                index   : in      Pcte.natural := Pcte.natural'FIRST;
                count   : in      Pcte.positive := Pcte.positive'LAST;
                status  : in      Pcte_error.handle := EXCEPTION_ONLY);

(12)        procedure copy (
                into_list     : in out  Pcte_oo.parameter_items.sequence;
                from_list     : in      Pcte_oo.parameter_items.sequence;
                into_index    : in      Pcte.natural := Pcte.natural'LAST;
                from_index    : in      Pcte.natural := Pcte.natural'FIRST;
                count         : in      Pcte.positive := Pcte.positive'LAST;
                status        : in      Pcte_error.handle := EXCEPTION_ONLY);

(13)        function length_of (
                list    : Pcte_oo.parameter_items.sequence;
                status  : Pcte_error.handle := EXCEPTION_ONLY)
                return    Pcte.natural;

(14)        function index_of (
                list    : Pcte_oo.parameter_items.sequence;
                item    : Pcte_oo.parameter_item;
                status  : Pcte_error.handle := EXCEPTION_ONLY)
                return    Pcte.integer;

(15)        function are_equal (
                first   : Pcte_oo.parameter_items.sequence;
                second  : Pcte_oo.parameter_items.sequence;
                status  : Pcte_error.handle := EXCEPTION_ONLY)
                return    Pcte.boolean;

(16)        procedure normalize (
                list    : in out  Pcte_oo.parameter_items.sequence;
                status  : in      Pcte_error.handle := EXCEPTION_ONLY);

(17)        procedure discard (
                list    : in out  Pcte_oo.parameter_items.sequence;
                status  : in      Pcte_error.handle := EXCEPTION_ONLY);

(18)    private

(19)        implementation-defined

(20)    end parameter_items;

(21)    type method_request is  record
            target_reference  : Pcte.object_reference;
            operation_id      : Pcte.type_reference;
            parameters        : Pcte_oo.parameter_items.sequence;
            context           : Pcte.object_reference;
        end record;

(22)    --      The semantics of the operations of this package are defined in 8.2.8 of ECMA-149.

(23)    package method_requests is

(24)        type sequence is limited private;

(25)        --      Pcte_oo.method_requests.sequence corresponds to the PCTE datatype
            --      Method_requests.
```

(26)
```
function get (
        list    : Pcte_oo.method_requests.sequence;
        index   : Pcte.natural := Pcte.natural'FIRST;
        status  : Pcte_error.handle := EXCEPTION_ONLY)
        return    Pcte.name;
```

(27)
```
procedure insert (
        list    : in out  Pcte_oo.method_requests.sequence;
        item    : in      Pcte_oo.method_request;
        index   : in      Pcte.natural := Pcte.natural'LAST;
        status  : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(28)
```
procedure replace (
        list    : in out  Pcte_oo.method_requests.sequence;
        item    : in      Pcte_oo.method_request;
        index   : in      Pcte.natural := Pcte.natural'LAST;
        status  : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(29)
```
procedure append (
        list    : in out  Pcte_oo.method_requests.sequence;
        item    : in      Pcte_oo.method_request;
        status  : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(30)
```
procedure delete (
        list    : in out  Pcte_oo.method_requests.sequence;
        index   : in      Pcte.natural := Pcte.natural'FIRST;
        count   : in      Pcte.positive := Pcte.positive'LAST;
        status  : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(31)
```
procedure copy (
        into_list       : in out  Pcte_oo.method_requests.sequence;
        from_list       : in      Pcte_oo.method_requests.sequence;
        into_index      : in      Pcte.natural := Pcte.natural'LAST;
        from_index      : in      Pcte.natural := Pcte.natural'FIRST;
        count           : in      Pcte.positive := Pcte.positive'LAST;
        status          : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(32)
```
function length_of (
        list    : Pcte_oo.method_requests.sequence;
        status  : Pcte_error.handle := EXCEPTION_ONLY)
        return    Pcte.natural;
```

(33)
```
function index_of (
        list    : Pcte_oo.method_requests.sequence;
        item    : Pcte_oo.method_request;
        status  : Pcte_error.handle := EXCEPTION_ONLY)
        return    Pcte.integer;
```

(34)
```
function are_equal (
        first   : Pcte_oo.method_requests.sequence;
        second  : Pcte_oo.method_requests.sequence;
        status  : Pcte_error.handle := EXCEPTION_ONLY)
        return    Pcte.boolean;
```

(35)
```
procedure normalize (
        list    : in out  Pcte_oo.method_requests.sequence;
        status  : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(36)
```
procedure discard (
        list    : in out  Pcte_oo.method_requests.sequence;
        status  : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(37) **private**

(38) *implementation-defined*

(39) **end** method_requests;

(40)
```
type context_adoption is (
        PCTE_ADOPT_WORKING_SCHEMA,
        PCTE_ADOPT_ACTIVITY,
        PCTE_ADOPT_USER,
        PCTE_ADOPT_OPEN_OBJECTS,
        PCTE_ADOPT_REFERENCE_OBJECTS,
        PCTE_ADOPT_ALL);
```

(41) --   The semantics of the operations of this package are defined in 8.2.8 of ECMA-149.

(42) **package** context_adoptions **is**

(43) **type** sequence **is limited private**;

(44)
```
--    Pcte_oo.context_adoptions.sequence corresponds to the PCTE datatype
--    Context_adoptions.
```

(45)
```
function get (
        list      : Pcte_oo.context_adoptions.sequence;
        index     : Pcte.natural := Pcte.natural'FIRST;
        status    : Pcte_error.handle := EXCEPTION_ONLY)
        return      Pcte.name;
```

(46)
```
procedure insert (
        list      : in out  Pcte_oo.context_adoptions.sequence;
        item      : in      Pcte_oo.context_adoption;
        index     : in      Pcte.natural := Pcte.natural'LAST;
        status    : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(47)
```
procedure replace (
        list      : in out  Pcte_oo.context_adoptions.sequence;
        item      : in      Pcte_oo.context_adoption;
        index     : in      Pcte.natural := Pcte.natural'LAST;
        status    : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(48)
```
procedure append (
        list      : in out  Pcte_oo.context_adoptions.sequence;
        item      : in      Pcte_oo.context_adoption;
        status    : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(49)
```
procedure delete (
        list      : in out  Pcte_oo.context_adoptions.sequence;
        index     : in      Pcte.natural := Pcte.natural'FIRST;
        count     : in      Pcte.positive := Pcte.positive'LAST;
        status    : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(50)
```
procedure copy (
        into_list         : in out  Pcte_oo.context_adoptions.sequence;
        from_list         : in      Pcte_oo.context_adoptions.sequence;
        into_index        : in      Pcte.natural := Pcte.natural'LAST;
        from_index        : in      Pcte.natural := Pcte.natural'FIRST;
        count             : in      Pcte.positive := Pcte.positive'LAST;
        status            : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(51)
```
function length_of (
        list      : Pcte_oo.context_adoptions.sequence;
        status    : Pcte_error.handle := EXCEPTION_ONLY)
        return      Pcte.natural;
```

```
(52)        function index_of (
                list    : Pcte_oo.context_adoptions.sequence;
                item    : Pcte_oo.context_adoption;
                status  : Pcte_error.handle := EXCEPTION_ONLY)
                return    Pcte.integer;

(53)        function are_equal (
                first   : Pcte_oo.context_adoptions.sequence;
                second  : Pcte_oo.context_adoptions.sequence;
                status  : Pcte_error.handle := EXCEPTION_ONLY)
                return    Pcte.boolean;

(54)        procedure normalize (
                list    : in out  Pcte_oo.context_adoptions.sequence;
                status  : in      Pcte_error.handle := EXCEPTION_ONLY);

(55)        procedure discard (
                list    : in out  Pcte_oo.context_adoptions.sequence;
                status  : in      Pcte_error.handle := EXCEPTION_ONLY);

(56)    private

(57)        implementation-defined

(58)    end context_adoptions;

(59)    type method_request_id is limited private;

(60)    --      The semantics of the operations of this package are defined in 8.2.8 of ECMA-149.

(61)    package method_request_ids is

(62)        type sequence is limited private;

(63)        --      Pcte_oo.method_request_ids.sequence corresponds to the PCTE datatype
            --      Method_request_ids.

(64)        function get (
                list    : Pcte_oo.method_request_ids.sequence;
                index   : Pcte.natural := Pcte.natural'FIRST;
                status  : Pcte_error.handle := EXCEPTION_ONLY)
                return    Pcte.name;

(65)        procedure insert (
                list            : in out  Pcte_oo.method_request_ids.sequence;
                item            : in      Pcte_oo.method_request_id;
                index           : in      Pcte.natural := Pcte.natural'LAST;
                status          : in      Pcte_error.handle := EXCEPTION_ONLY);

(66)        procedure replace (
                list            : in out  Pcte_oo.method_request_ids.sequence;
                item            : in      Pcte_oo.method_request_id;
                index           : in      Pcte.natural := Pcte.natural'LAST;
                status          : in      Pcte_error.handle := EXCEPTION_ONLY);

(67)        procedure append (
                list            : in out  Pcte_oo.method_request_ids.sequence;
                item            : in      Pcte_oo.method_request_id;
                status          : in      Pcte_error.handle := EXCEPTION_ONLY);

(68)        procedure delete (
                list            : in out  Pcte_oo.method_request_ids.sequence;
                index           : in      Pcte.natural := Pcte.natural'FIRST;
                count           : in      Pcte.positive := Pcte.positive'LAST;
                status          : in      Pcte_error.handle := EXCEPTION_ONLY);
```

(69)
```
        procedure copy (
            into_list       : in out  Pcte_oo.method_request_ids.sequence;
            from_list       : in      Pcte_oo.method_request_ids.sequence;
            into_index      : in      Pcte.natural := Pcte.natural'LAST;
            from_index      : in      Pcte.natural := Pcte.natural'FIRST;
            count           : in      Pcte.positive := Pcte.positive'LAST;
            status          : in      Pcte_error.handle := EXCEPTION_ONLY);
```
(70)
```
        function length_of (
            list    : Pcte_oo.method_request_ids.sequence;
            status  : Pcte_error.handle := EXCEPTION_ONLY)
            return    Pcte.natural;
```
(71)
```
        function index_of (
            list    : Pcte_oo.method_request_ids.sequence;
            item    : Pcte_oo.method_request_id;
            status  : Pcte_error.handle := EXCEPTION_ONLY)
            return    Pcte.integer;
```
(72)
```
        function are_equal (
            first   : Pcte_oo.method_request_ids.sequence;
            second  : Pcte_oo.method_request_ids.sequence;
            status  : Pcte_error.handle := EXCEPTION_ONLY)
            return    Pcte.boolean;
```
(73)
```
        procedure normalize (
            list    : in out  Pcte_oo.method_request_ids.sequence;
            status  : in      Pcte_error.handle := EXCEPTION_ONLY);
```
(74)
```
        procedure discard (
            list    : in out  Pcte_oo.method_request_ids.sequence;
            status  : in      Pcte_error.handle := EXCEPTION_ONLY);
```
(75)
```
    private
```
(76)
```
        implementation-defined
```
(77)
```
    end method_request_ids;
```

9.2 Object-oriented invocation management operations

(1)
```
    package process is

    -- 9.2.1 PROCESS_ADOPT_CONTEXT
```
(2)
```
        procedure adopt_context (
            context_adoptions           : in Pcte_oo.context_adoptions.sequence)
```
(3)
```
    end process;
```
(4)
```
    package request is

    -- 9.2.2 REQUEST_INVOKE
```
(5)
```
        function invoke (
            request                     : Pcte_oo.method_request;
            context_adoptions           : Pcte_oo.context_adoptions)
            return Pcte_oo.method_request_id;

    -- 9.2.3 REQUEST_SEND
```
(6)
```
        function send (
            request                     : Pcte_oo.method_request;
            context_adoptions           : Pcte_oo.context_adoptions.sequence)
            return Pcte_oo.method_request_id;
        -- 9.2.4 REQUEST_SEND_MULTIPLE

(7)             function send_multiple (
                    requests                : Pcte_oo.method_requests.sequence;
                    context_adoptions       : Pcte_oo.context_adoptions.sequence)
                    return Pcte_oo.method_request_ids.sequence;

(8)     end request;
```

10 Object-oriented schema management

```
(1)     package sds is
```

10.1 Object-oriented schema management datatypes

```
(1)             type interface_scope is (NO_OPERATION, ALL_OPERATIONS);
```

10.2 Object-oriented schema management operations

```
        -- 10.2.1 SDS_APPLY_INTERFACE_TYPE

(1)             procedure apply_interface_type (
                    sds             : in Pcte.object_reference;
                    interface_type  : in Pcte.type_reference;
                    type            : in Pcte.type_reference);

        -- 10.2.2 SDS_APPLY_OPERATION_TYPE

(2)             procedure apply_operation_type (
                    sds             : in Pcte.object_reference;
                    operation_type  : in Pcte.type_reference;
                    type            : in Pcte.type_reference);

        -- 10.2.3 SDS_CREATE_DATA_PARAMETER_TYPE

(3)             function create_data_parameter_type (
                    sds             : Pcte.object_reference;
                    local_name      : Pcte.name := "";
                    data_type       : Pcte.type_reference)
                    return  Pcte.type_reference;

        -- 10.2.4 SDS_CREATE_INTERFACE_PARAMETER_TYPE

(4)             function create_interface_parameter_type (
                    sds             : Pcte.object_reference;
                    local_name      : Pcte.name := "";
                    interface_type  : Pcte.type_reference)
                    return  Pcte.type_reference;

        -- 10.2.5 SDS_CREATE_INTERFACE_TYPE

(5)             function create_interface_type (
                    sds             : Pcte.object_reference;
                    local_name      : Pcte.name := "";
                    parents         : Pcte.type_references.sequence;
                    new_operations  : Pcte.type_references.sequence)
                    return Pcte.type_reference;

        -- 10.2.6 SDS_CREATE_OBJECT_PARAMETER_TYPE

(6)             function create_data_parameter_type (
                    sds             : Pcte.object_reference;
                    local_name      : Pcte.name := "";
                    object_type     : Pcte.type_reference)
                    return  Pcte.type_reference;
        -- 10.2.7 SDS_CREATE_OPERATION_TYPE
(7)             function create_operation_type (
                        sds             : Pcte.object_reference;
                        local_name      : Pcte.name := "";
                        parameters      : Pcte.type_references.sequence;
                        return_value    : Pcte.type_reference)
                        return  Pcte.type_reference;

        -- 10.2.8 SDS_IMPORT_INTERFACE_TYPE
(8)             procedure import_interface_type (
                        to_sds          : in Pcte.object_reference;
                        from_sds        : in Pcte.object_reference;
                        type            : in Pcte.type_reference;
                        local_name      : in Pcte.name := ""
                        import_scope    : in Pcte_oo.sds.interface_scope);

        -- 10.2.9 SDS_IMPORT_OPERATION_TYPE
(9)             procedure import_operation_type (
                        to_sds          : in Pcte.object_reference;
                        from_sds        : in Pcte.object_reference;
                        type            : in Pcte.type_reference;
                        local_name      : in Pcte.name := "");

        -- 10.2.10 SDS_UNAPPLY_INTERFACE_TYPE
(10)            procedure unapply_interface_type (
                        sds             : in Pcte.object_reference;
                        interface_type  : in Pcte.type_reference;
                        type            : in Pcte.type_reference);

        -- 10.2.11 SDS_UNAPPLY_OPERATION_TYPE
(11)            procedure unapply_operation_type (
                        sds             : in Pcte.object_reference;
                        operation_type  : in Pcte.type_reference;
                        type            : in Pcte.type_reference);

(12)    end sds;

(13) end Pcte_oo;
```

11 Error conditions

(1) The following new values are added to the enumeration type Pcte_error.error_code defined in clause 25 of ECMA-162, with the code letters shown. For upward compatibility from ECMA-162 they are added at the end, after the Ada-binding-defined errors.

```
(2) NUMBER_OF_PARAMETERS_IS_WRONG,
(3) OPERATION_METHOD_NOT_FOUND,                          -- U
(4) OPERATION_METHOD_COULD_NOT_BE_ACTIVATED,             -- U
(5) TYPE_IS_ALREADY_CONSTRAINED,                         -- U
(6) TYPE_OF_PARAMETER_IS_WRONG,                          -- U
```