
**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)



Contents

1 Scope	1
2 Conformance	1
3 Normative references	1
4 Definitions	2
5 Formal notations	2
6 Outline of the Standard	2
7 Binding strategy	2
7.1 IDL standard	2
7.2 General principles	2
7.3 Sets and sequences	3
7.4 References and names	3
7.5 Implementation aspects	4
7.5.1 Source files	4
7.5.2 Naming changes in the IDL	4
7.5.3 Difference in generated C code	4
8 Datatype mapping	4
8.1 Basic datatypes	4
8.2 Sequences	5
8.3 The global pcte source file	8
8.4 The PCTE basic type source file	9
9 Object management	9
9.1 Object management datatypes	9
9.2 Link operators	12
9.3 Object operations	16
9.4 Version operations	20
9.5 Object and version operations – reference interfaces	21
10 Schema management	25
10.1 Schema management datatypes	25
10.2 Update operations	26

© ISO/IEC 1998

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 the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

10.3 Usage operations	32
10.4 Working schema operations	35
11 Volumes, devices, archives, and clusters	39
11.1 Volume, device, archive, and cluster datatypes	39
11.2 Volume, device, and archive operations	40
11.3 Cluster operations	43
12 Files, pipes, and devices	44
12.1 File, pipe, and device datatypes	44
12.2 File, pipe, and device operations	44
13 Process execution	47
13.1 Process execution datatypes	47
13.2 Process execution operations	48
13.3 Security operations	51
13.4 Profiling operations	52
13.5 Monitoring operations	53
13.6 Mandatory security operations	54
13.7 Consumer identity operations	54
13.8 Contents handle operation	54
14 Message queues	55
14.1 Message queue datatypes	55
14.2 Message queue operations	56
15 Notification	58
15.1 Notification datatypes	58
15.2 Notification operations	58
16 Concurrency and integrity control	59
16.1 Concurrency and integrity control datatypes	59
16.2 Concurrency and integrity control operations	60
17 Replication	61
17.1 Replication datatypes	61
17.2 Replication operations	61
18 Network connection	63
18.1 Network connection datatypes	63
18.2 Network connection operations	64
18.3 Foreign system operations	65
18.4 Time operations	65
18.5 Other workstation operations	66
19 Discretionary security	66
19.1 Discretionary security datatypes	67
19.2 Discretionary access control operations	67
19.3 Discretionary security administration operations	68

20 Mandatory security	70
20.1 Mandatory_security datatypes	70
20.2 Operations for mandatory security operation	71
20.3 Mandatory security administration operations	71
21 Auditing	73
21.1 Auditing datatypes	73
21.2 Auditing operations	77
22 Accounting	78
22.1 Accounting datatypes	78
22.2 Accounting administration operations	80
23 References	82
23.1 Reference datatypes	82
23.2 Reference creation and discarding	83
23.3 Object reference operations	84
23.4 Link reference operations	85
23.5 Type reference operations	86
24 Implementation limits	87
24.1 Implementation limit datatypes	87
24.2 Implementation limit operations	89
25 Error conditions	89
25.1 Error condition datatypes	89
Annex A - Comparison with ISO/IEC 13719-2	97
Annex B - IDL file structure	100
Annex C - The object-oriented module	103
Index of abstract operations	108
Index of IDL subprograms	109
Index of IDL datatypes	125

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,	<i>Information technology - Portable Common Tool Environment (PCTE) - Part 1: Abstract specification.</i>
ISO/IEC 13719-2:1998,	<i>Information technology - Portable Common Tool Environment (PCTE) - Part 2: C programming language binding.</i>
ISO/IEC 14750:— ¹⁾ ,	<i>Information technology - Open Distributed Processing - Interface Definition Language.</i>

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_DESCRIPTOR_EI, PCTE_H_LINK_SET_DESCRIPTOR_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_octetarray;
};

#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 ;
```

```

#define PCTE_ALL_CATEGORIES (Pcte_natural) PCTE_COMPOSITION |\
    PCTE_EXISTENCE |\
    PCTE_REFERENCE |\
    PCTE_DESIGNATION |\
    PCTE_IMPLICIT

enum Pcte_value_type {
    PCTE_BOOLEAN_ATTRIBUTE,
    PCTE_INTEGER_ATTRIBUTE,
    PCTE_NATURAL_ATTRIBUTE,
    PCTE_FLOAT_ATTRIBUTE,
    PCTE_STRING_ATTRIBUTE,
    PCTE_TIME_ATTRIBUTE,
    PCTE_ENUMERATION_ATTRIBUTE
};

union Pcte_value_value switch (long) {
    case 1 : Pcte_boolean    v_boolean;
    case 2 : Pcte_integer    v_integer;
    case 3 : Pcte_natural    v_natural;
    case 4 : Pcte_float      v_float;
    case 5 : Pcte_string     v_string;
    case 6 : Pcte_time       v_time;
    case 7 : Pcte_natural    v_enumeral_type_position;
};

struct Pcte_attribute_value {
    Pcte_value_type    type;
    Pcte_value_value   value;
};

struct Pcte_attribute_assignment {
    Pcte_attribute_name    name;
    Pcte_attribute_value   value;
};

struct Pcte_h_attribute_assignment {
    Pcte_attribute_reference    reference;
    Pcte_attribute_value       value;
};

enum Pcte_link_scope {
    PCTE_INTERNAL_LINKS, PCTE_EXTERNAL_LINKS, PCTE_ALL_LINKS
};

enum Pcte_type_ancestry {
    PCTE_EQUAL_TYPE, PCTE_ANCESTOR_TYPE,
    PCTE_DESCENDANT_TYPE, PCTE_UNRELATED_TYPE
};

```

```
enum Pcte_version_relation {
    PCTE_ANCESTOR_VSN, PCTE_DESCENDANT_VSN, PCTE_SAME_VSN,
    PCTE_RELATED_VSN, PCTE_UNRELATED_VSN
};

enum Pcte_object_scope {
    PCTE_ATOMIC, PCTE_COMPOSITE
};

#define PCTE_MAX_EXACT_IDENTIFIER_SIZE PCTE_MAX_KEY_SIZE

typedef Pcte_octet
    Pcte_exact_identifier [PCTE_MAX_EXACT_IDENTIFIER_SIZE + 1];

#endif    // !PCTE_TYPES_INCLUDED

/* The source file "oms.idl" */

#ifndef PCTE_OMS_INCLUDED
#define PCTE_OMS_INCLUDED 1

#include "types.idl"
#include "references.idl"
#include "oms_types.idl"
#include "sequences.idl"
#include "contents_types.idl"

typedef Object Pcte_contents;

typedef Pcte_sequence Pcte_attribute_assignments;

typedef Pcte_sequence Pcte_h_attribute_assignments;

enum Pcte_volume_accessibility {
    PCTE_ACCESSIBLE, PCTE_INACCESSIBLE, PCTE_UNKNOWN
};

#include "devices.idl"

struct Pcte_volume_info {
    Pcte_volume_identifier    volume;
    Pcte_volume_accessibility mounted;
};

struct Pcte_link_set_descriptor {
    Pcte_object_reference    origin;
    Pcte_link_names          links;
};

struct Pcte_h_link_set_descriptor {
    Pcte_object_reference    origin;
    Pcte_link_references     links;
};

#include "discretionary.idl"
```

9.2 Link operators

```
interface Pcte_link {  
    /* This interface is applied to the PCTE object type "object" */  
  
    /* 9.2.1 LINK_CREATE */  
    Pcte_error_type create (  
        in Pcte_link_name      new_link,  
        in Pcte_object_reference dest,  
        in Pcte_key            reverse_key  
    );  
  
    /* 9.2.2 LINK_DELETE */  
    Pcte_error_type delete (  
        in Pcte_link_name link  
    );  
  
    /* 9.2.3 LINK_DELETE_ATTRIBUTE */  
    Pcte_error_type delete_attribute (  
        in Pcte_link_name      link,  
        in Pcte_attribute_reference attribute_ref  
    );  
  
    /* 9.2.4 LINK_GET_ATTRIBUTE */  
    Pcte_error_type get_attribute (  
        in Pcte_link_name      link,  
        in Pcte_attribute_name  name,  
        out Pcte_attribute_value value  
    );  
  
    /* 9.2.5 LINK_GET_DESTINATION_VOLUME */  
    Pcte_error_type get_destination_volume (  
        in Pcte_link_name      link,  
        out Pcte_volume_info    volume_info  
    );  
  
    /* 9.2.6 LINK_GET_KEY */  
    Pcte_error_type get_key (  
        in Pcte_link_name link,  
        out Pcte_key      key  
    );  
};
```

```
/* 9.2.7 LINK_GET_REVERSE */
Pcte_error_type get_reverse (
    in Pcte_link_name      link,
    out Pcte_link_name      reverse_link,
    out Pcte_object_reference dest
);

/* 9.2.8 LINK_GET_SEVERAL_ATTRIBUTES */
Pcte_error_type get_attributes_in_working_schema (
    in Pcte_link_name      link,
    out Pcte_attribute_assignments values
);

Pcte_error_type get_attributes_of_types (
    in Pcte_link_name      link,
    in Pcte_attribute_names attributes,
    out Pcte_attribute_assignments values
);

/* 9.2.9 LINK_REPLACE */
Pcte_error_type replace (
    in Pcte_link_name      link,
    in Pcte_object_reference new_origin,
    in Pcte_link_name      new_link,
    in Pcte_key             new_reverse_key
);

/* 9.2.10 LINK_RESET_ATTRIBUTE */
Pcte_error_type reset_attribute (
    in Pcte_link_name      link,
    in Pcte_attribute_reference attribute_ref
);

/* 9.2.11 LINK_SET_ATTRIBUTE */
Pcte_error_type set_attribute (
    in Pcte_link_name      link,
    in Pcte_attribute_name attribute_ref,
    in Pcte_attribute_value value
);

/* 9.2.12 LINK_SET_SEVERAL_ATTRIBUTES */
Pcte_error_type set_several_attributes (
    in Pcte_link_name      link,
    in Pcte_attribute_assignments attributes
);
```

```

/* 11.2.7 LINK_GET_DESTINATION_ARCHIVE */
Pcte_error_type get_destination_archive (
    in Pcte_link_name      link,
    out Pcte_archive_identifier archive_identifier
);
};

interface Pcte_h_link {
/* This interface is applied to the PCTE object type "object" \f B*/

/* 9.2.1 LINK_CREATE */
Pcte_error_type create (
    in Pcte_link_reference    new_link,
    in Pcte_object_reference  dest,
    in Pcte_key               reverse_key
);
/* 9.2.2 LINK_DELETE */
Pcte_error_type delete (
    in Pcte_link_reference  link
);
/* 9.2.3 LINK_DELETE_ATTRIBUTE */
Pcte_error_type delete_attribute (
    in Pcte_link_reference    link,
    in Pcte_attribute_reference attribute_ref
);
/* 9.2.4 LINK_GET_ATTRIBUTE */
Pcte_error_type get_attribute (
    in Pcte_link_reference    link,
    in Pcte_attribute_reference attribute_ref,
    out Pcte_attribute_value  value
);
/* 9.2.5 LINK_GET_DESTINATION_VOLUME */
Pcte_error_type get_destination_volume (
    in Pcte_link_reference  link,
    out Pcte_volume_inf    volume_info
);
/* 9.2.6 LINK_GET_KEY */
Pcte_error_type get_key (
    in Pcte_link_reference  link,
    out Pcte_key            key
);
};

```

```
/* 9.2.7 LINK_GET_REVERSE */
Pcte_error_type get_reverse (
    in Pcte_link_reference      link,
    out Pcte_link_reference     reverse_link,
    out Pcte_object_reference   dest
);

/* 9.2.8 LINK_GET_SEVERAL_ATTRIBUTES */
Pcte_error_type get_attributes_in_working_schema (
    in Pcte_link_reference      link,
    out Pcte_h_attribute_assignments values
);

Pcte_error_type get_attributes_of_types (
    in Pcte_link_reference      link,
    in Pcte_attribute_references attributes,
    out Pcte_h_attribute_assignments values
);

/* 9.2.9 LINK_REPLACE */
Pcte_error_type replace (
    in Pcte_link_reference      link,
    in Pcte_object_reference    new_origin,
    in Pcte_link_reference      new_link,
    in Pcte_key                 new_reverse_key
);

/* 9.2.10 LINK_RESET_ATTRIBUTE */
Pcte_error_type reset_attribute (
    in Pcte_link_reference      link,
    in Pcte_attribute_reference attribute_ref
);

/* 9.2.11 LINK_SET_ATTRIBUTE */
Pcte_error_type set_attribute (
    in Pcte_link_reference      link,
    in Pcte_attribute_reference attribute_ref,
    out Pcte_attribute_value    value
);

/* 9.2.12 LINK_SET_SEVERAL_ATTRIBUTES */
Pcte_error_type set_several_attributes (
    in Pcte_link_reference      link,
    in Pcte_h_attribute_assignments attributes
);
```

```

/* 11.2.7 LINK_GET_DESTINATION_ARCHIVE */
Pcte_error_type get_destination_archive (
    in Pcte_link_reference      link,
    out Pcte_archive_identifier archive_identifier
);
};

```

9.3 Object operations

```

interface Pcte_object {
/* This interface is applied to the PCTE object type "object" */

/* 9.3.1 OBJECT_CHECK_TYPE */
Pcte_error_type check_type (
    in Pcte_type_name      type2,
    in Pcte_type_ancestry relation
);

/* 9.3.2 OBJECT_CONVERT */
Pcte_error_type convert (
    in Pcte_type_name      type
);

/* 9.3.3 OBJECT_COPY */
Pcte_error_type copy (
    in Pcte_link_name      new_link,
    in Pcte_key             reverse_key,
    in Pcte_object_reference on_same_volume_as,
    in Pcte_atomic_access_rights access_mask,
    out Pcte_object_reference new_object
);

/* 9.3.4 OBJECT_CREATE */
Pcte_error_type create (
    in Pcte_type_name      type,
    in Pcte_link_name      new_link,
    in Pcte_key             reverse_key,
    in Pcte_object_reference on_same_volume_as,
    in Pcte_atomic_access_rights access_mask,
    out Pcte_object_reference new_object
);

/* 9.3.5 OBJECT_DELETE */
Pcte_error_type delete (
    in Pcte_link_name      link
);

```

```
/* 9.3.6 OBJECT_DELETE_ATTRIBUTE */
Pcte_error_type delete_attribute (
    in Pcte_attribute_name  attribute_ref
);

/* 9.3.7 OBJECT_GET_ATTRIBUTE */
Pcte_error_type get_attribute (
    in Pcte_attribute_name  attribute_ref,
    out Pcte_attribute_value value
);

/* 9.3.8 OBJECT_GET_PREFERENCE */
Pcte_error_type get_preference (
    out Pcte_key            key,
    out Pcte_type_name      type
);

/* 9.3.9 OBJECT_GET_SEVERAL_ATTRIBUTES */
Pcte_error_type get_attributes_in_working_schema (
    out Pcte_attribute_assignments values
);

Pcte_error_type get_attributes_of_types (
    in Pcte_attribute_names      attributes,
    out Pcte_attribute_assignments values
);

/* 9.3.10 OBJECT_GET_TYPE */
Pcte_error_type get_type (
    out Pcte_type_name          type
);

/* 9.3.11 OBJECT_IS_COMPONENT */
Pcte_error_type is_component (
    in Pcte_object_referencecomponent,
    out Pcte_boolean             value
);

/* 9.3.12 OBJECT_LIST_LINKS */
Pcte_error_type list_all_links (
    in Pcte_link_scope          extent,
    in Pcte_object_scope        scope,
    in Pcte_categories           categories,
    out Pcte_link_set_descriptors links
);
```

```

Pcte_error_type list_links_in_working_schema (
    in Pcte_link_scope          extent,
    in Pcte_object_scope        scope,
    in Pcte_categories          categories,
    out Pcte_link_set_descriptors links
);

Pcte_error_type list_links_of_types (
    in Pcte_link_scope          extent,
    in Pcte_object_scope        scope,
    in Pcte_type_names          types,
    out Pcte_link_set_descriptors links
);

/* 9.3.13 OBJECT_LIST_VOLUMES */
Pcte_error_type list_volumes (
    out Pcte_volume_infos volumes
);

/* 9.3.14 OBJECT_MOVE */
Pcte_error_type move (
    in Pcte_object_reference    on_same_volume_as,
    in Pcte_object_scope        scope
);

/* 9.3.15 OBJECT_RESET_ATTRIBUTE */
Pcte_error_type reset_attribute (
    in Pcte_attribute_name      attribute_ref
);

/* 9.3.16 OBJECT_SET_ATTRIBUTE */
Pcte_error_type set_attribute (
    in Pcte_attribute_name      attribute_ref,
    in Pcte_attribute_value      value
);

/* 9.3.17 OBJECT_SET_PREFERENCE */
Pcte_error_type set_preference (
    in Pcte_type_name          type,
    in Pcte_key                key
);

/* 9.3.18 OBJECT_SET_SEVERAL_ATTRIBUTES */
Pcte_error_type set_several_attributes (
    in Pcte_attribute_assignments attributes
);

```

```

/* 9.3.19 OBJECT_SET_TIME_ATTRIBUTES */
Pcte_error_type set_time_attributes (
    in Pcte_time          last_access,
    in Pcte_time          last_modification,
    in Pcte_object_scope  scope
);

/* 9.3.20 VOLUME_LIST_OBJECTS */
/* See 11.2. */

/* 20.2.5 OBJECT_SET_CONFIDENTIALITY_LABEL */
Pcte_error_type set_confidentiality_label (
    in Pcte_security_label  label
);

/* 20.2.6 OBJECT_SET_INTEGRITY_LABEL */
Pcte_error_type set_integrity_label (
    in Pcte_security_label  label
);

/* 19.2.2 OBJECT_CHECK_PERMISSION */
Pcte_error_type check_permission (
    in Pcte_discretionary_access_modes  modes,
    in Pcte_object_scope                scope,
    out Pcte_boolean                    accessible
);

/* 19.2.3 OBJECT_GET_ACL */
Pcte_error_type get_acl (
    in Pcte_object_scope  scope,
    out Pcte_acl          acl
);

/* 19.2.4 OBJECT_SET_ACL_ENTRY */
Pcte_error_type set_acl_entry (
    in Pcte_group_identifier  group,
    in Pcte_requested_access_rights  modes,
    in Pcte_object_scope          scope
);

/* 11.2.1 ARCHIVE_CREATE */
Pcte_error_type archive_create (
    in Pcte_natural          archive_identifier,
    in Pcte_object_reference on_same_volume_as,
    out Pcte_atomic_access_rights  access_mask,
    out Pcte_object_reference      new_archive
);

```

```

/* 12.2.6 CONTENTS_OPEN */
Pcte_error_type contents_open (
    in Pcte_contents_access_mode    opening_mode,
    in Pcte_boolean                  non_blocking_io,
    in Pcte_boolean                  inheritable,
    out Pcte_contents                contents
);
};

```

9.4 Version operations

```

interface Pcte_version {
/* This interface is applied to the PCTE object type "object". */

/* 9.4.1 VERSION_ADD_PREDECESSOR */
Pcte_error_type add_predecessor (
    in Pcte_object_reference    new_predecessor
);

/* 9.4.2 VERSION_IS_CHANGED */
Pcte_error_type is_changed (
    in Pcte_key                predecessor,
    out Pcte_boolean           changed
);

/* 9.4.3 VERSION_REMOVE */
Pcte_error_type remove (
);

/* 9.4.4 VERSION_REMOVE_PREDECESSOR */
Pcte_error_type remove_predecessor (
    in Pcte_object_reference    predecessor
);

/* 9.4.5 VERSION_REVISE */
Pcte_error_type revise (
    in Pcte_object_reference    new_origin,
    in Pcte_link_name           new_link,
    in Pcte_object_reference    on_same_volume_as,
    in Pcte_atomic_access_rights access_mask,
    out Pcte_object_reference    new_version
);
};

```

```

/* 9.4.6 VERSION_SNAPSHOT */
Pcte_error_type snapshot (
    in Pcte_object_reference    new_origin,
    in Pcte_link_name          new_link,
    in Pcte_object_reference    on_same_volume_as,
    in Pcte_atomic_access_rights access_mask,
    out Pcte_object_reference    new_version
);

/* 9.4.7 VERSION_TEST_ANCESTRY */
Pcte_error_type test_ancestry (
    in Pcte_object_reference    version2,
    out Pcte_version_relation    ancestry
);

/* 9.4.8 VERSION_TEST_DESCENT */
Pcte_error_type test_descent (
    in Pcte_object_reference    version2,
    out Pcte_version_relation    descent
);
};

```

9.5 Object and version operations – reference interfaces

```

interface Pcte_h_object {
    /* This interface is applied to the PCTE object type "object". */

    /* 9.3.1 OBJECT_CHECK_TYPE */
    Pcte_error_type check_type (
        in Pcte_type_reference    type2,
        in Pcte_type_ancestry    relation
    );

    /* 9.3.2 OBJECT_CONVERT */
    Pcte_error_type convert (
        in Pcte_type_reference    type
    );
}

```

/* 9.3.3 OBJECT_COPY */

```
Pcte_error_type copy (
    in Pcte_object_reference    new_origin,
    in Pcte_link_reference      new_link,
    in Pcte_key                 reverse_key,
    in Pcte_object_reference    on_same_volume_as,
    in Pcte_atomic_access_rights access_mask,
    out Pcte_object_reference    new_object
);
```

/* 9.3.4 OBJECT_CREATE */

```
Pcte_error_type create (
    in Pcte_type_reference      type,
    in Pcte_link_reference      new_link,
    in Pcte_key                 reverse_key,
    in Pcte_object_reference    on_same_volume_as,
    in Pcte_atomic_access_rights access_mask,
    out Pcte_object_reference    new_object
);
```

/* 9.3.5 OBJECT_DELETE */

```
Pcte_error_type delete (
    in Pcte_link_reference      link
);
```

/* 9.3.6 OBJECT_DELETE_ATTRIBUTE */

```
Pcte_error_type delete_attribute (
    in Pcte_attribute_reference attribute_ref
);
```

/* 9.3.7 OBJECT_GET_ATTRIBUTE */

```
Pcte_error_type get_attribute (
    in Pcte_attribute_reference attribute_ref,
    out Pcte_attribute_value    value
);
```

/* 9.3.8 OBJECT_GET_PREFERENCE */

```
Pcte_error_type get_preference (
    out Pcte_key                key,
    out Pcte_link_reference      type
);
```

/* 9.3.9 OBJECT_GET_SEVERAL_ATTRIBUTES */

```
Pcte_error_type get_attributes_in_working_schema (
    out Pcte_h_attribute_assignments values
);
```

```
Pcte_error_type get_attributes_of_types (  
    in Pcte_attribute_references    attributes,  
    out Pcte_h_attribute_assignments values  
);
```

```
/* 9.3.10 OBJECT_GET_TYPE */
```

```
Pcte_error_type get_type (  
    out Pcte_type_reference    type  
);
```

```
/* 9.3.12 OBJECT_LIST_LINKS */
```

```
Pcte_error_type list_all_links (  
    in Pcte_link_scope          extent,  
    in Pcte_object_scope        scope,  
    in Pcte_categories          categories,  
    out Pcte_h_link_set_descriptors links  
);
```

```
Pcte_error_type list_links_in_working_schema (  
    in Pcte_link_scope          extent,  
    in Pcte_object_scope        scope,  
    in Pcte_categories          categories,  
    out Pcte_h_link_set_descriptors links  
);
```

```
Pcte_error_type list_links_of_types (  
    in Pcte_link_scope          extent,  
    in Pcte_object_scope        scope,  
    in Pcte_type_references     types,  
    out Pcte_h_link_set_descriptors links  
);
```

```
/* 9.3.15 OBJECT_RESET_ATTRIBUTE */
```

```
Pcte_error_type reset_attribute (  
    in Pcte_attribute_reference attribute_ref  
);
```

```
/* 9.3.16 OBJECT_SET_ATTRIBUTE */
```

```
Pcte_error_type set_attribute (  
    in Pcte_attribute_reference attribute_ref,  
    in Pcte_attribute_value    value  
);
```

```
/* 9.3.17 OBJECT_SET_PREFERENCE */
```

```
Pcte_error_type set_preference (  
    in Pcte_type_reference type,  
    in Pcte_key            key  
);
```

```

/* 9.3.18 OBJECT_SET_SEVERAL_ATTRIBUTES */
Pcte_error_type set_several_attributes (
    in Pcte_h_attribute_assignments  attributes
);

/* 11.2.1 ARCHIVE_CREATE */
Pcte_error_type archive_create (
    in Pcte_natural          archive_identifier,
    in Pcte_object_reference  on_same_volume_as,
    out Pcte_atomic_access_rights access_mask,
    out Pcte_object_reference  new_archive
);

/* 12.2.6 CONTENTS_OPEN */
Pcte_error_type contents_open (
    in Pcte_contents_access_mode  opening_mode,
    in Pcte_boolean               non_blocking_io,
    in Pcte_boolean               inheritable,
    out Pcte_contents             contents
);
};

interface Pcte_h_version {
/* This interface is applied to the PCTE object type "object" \f B. */

/* 9.4.5 VERSION_REVISION */
Pcte_error_type revise (
    in Pcte_object_reference  new_origin,
    in Pcte_link_reference    new_link,
    in Pcte_object_reference  on_same_volume_as,
    in Pcte_atomic_access_rights access_mask,
    out Pcte_object_reference  new_version
);

/* 9.4.6 VERSION_SNAPSHOT */
Pcte_error_type snapshot (
    in Pcte_object_reference  new_origin,
    in Pcte_link_reference    new_link,
    in Pcte_object_reference  on_same_volume_as,
    in Pcte_atomic_access_rights access_mask,
    out Pcte_object_reference  new_version
);
};

#endif

```

10 Schema management

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

#ifndef PCTE_SMS_INCLUDED
#define PCTE_SMS_INCLUDED 1

#include "types.idl"
#include "references.idl"
#include "sequences.idl"#include "oms_types.idl"
```

10.1 Schema management datatypes

```
enum Pcte_definition_mode_value {
    PCTE_CREATE_MODE,
    PCTE_DELETE_MODE,
    PCTE_READ_MODE,
    PCTE_WRITE_MODE,
    PCTE_NAVIGATE_MODE
};

typedef Pcte_natural Pcte_definition_mode_values;

enum Pcte_duplication {
    PCTE_DUPLICATED, PCTE_NOT_DUPLICATED
};

enum Pcte_exclusiveness {
    PCTE_SHARABLE, PCTE_EXCLUSIVE
};

enum Pcte_stability {
    PCTE_ATOMIC_STABLE, PCTE_COMPOSITE_STABLE, PCTE_NOT_STABLE
};

enum Pcte_contents_type {
    PCTE_NO_CONTENTS, PCTE_FILE_TYPE,
    PCTE_PIPE_TYPE, PCTE_DEVICE_TYPE,
    PCTE_AUDIT_FILE_TYPE, PCTE_ACCOUNTING_LOG_TYPE
};

/* Pcte_contents_type corresponds to the PCTE datatype Contents_type. The value */
/* PCTE_NO_CONTENTS corresponds to the absence of a Contents_type result from */
/* SDS_GET_OBJECT_TYPE_PROPERTIES and */
/* WS_GET_OBJECT_TYPE_PROPERTIES. */
struct Pcte_link_flags {
    Pcte_category      category;
    Pcte_stability     stability;
    Pcte_exclusiveness exclusiveness;
    Pcte_duplication   duplication;
};
```

```

struct Pcte_link_type_properties {
    Pcte_link_flags  link_type_flag;
    Pcte_natural     lower_bound, upper_bound;
};

/* Pcte_link_type_properties corresponds to a number of parameter types in */
/* SDS_CREATE_RELATIONSHIP_TYPE, and to a number of result types of */
/* SDS_GET_LINK_TYPE_PROPERTIES and */
/* WS_GET_LINK_TYPE_PROPERTIES. */
enum Pcte_attribute_scan_kind {
    PCTE_OBJECT, PCTE_OBJECT_ALL, PCTE_LINK_KEY, PCTE_LINK_NON_KEY
};
enum Pcte_link_scan_kind {
    PCTE_ORIGIN, PCTE_ORIGIN_ALL, PCTE_DESTINATION,
    PCTE_DESTINATION_ALL, PCTE_KEY_ATTR, PCTE_NON_KEY_ATTR
};
enum Pcte_object_scan_kind {
    PCTE_CHILD, PCTE_DESCENDANT, PCTE_PARENT, PCTE_ANCESTOR,
    PCTE_ATTRIBUTE, PCTE_ATTRIBUTE_ALL, PCTE_LINK_ORIGIN,
    PCTE_LINK_ORIGIN_ALL, PCTE_LINK_DESTINATION,
    PCTE_LINK_DESTINATION_ALL
};
enum Pcte_type_kind {
    PCTE_OBJECT_TYPE, PCTE_LINK_TYPE, PCTE_ATTRIBUTE_TYPE,
    PCTE_ENUMERAL_TYPE
};
#define PCTE_MAX_ENUMERAL_TYPE_IMAGE_SIZE PCTE_MAX_NAME_SIZE
typedef Pcte_octet Pcte_enumeral_type_image
    [PCTE_MAX_ENUMERAL_TYPE_IMAGE_SIZE + 1];

```

10.2 Update operations

```

interface Pcte_sds {

    /* This interface is applied to the PCTE object type "sds". */

    /* 10.2.1 SDS_ADD_DESTINATION */

    Pcte_error_type add_destination (
        in Pcte_type_name_in_sds link_type,
        in Pcte_type_name_in_sds object_type
    );

```

```
/* 10.2.2 SDS_APPLY_ATTRIBUTE_TYPE */
```

```
Pcte_error_type apply_attribute_type (
    in Pcte_type_name_in_sds attribute_type,
    in Pcte_type_name_in_sds type
);
```

```
/* 10.2.3 SDS_APPLY_LINK_TYPE */
```

```
Pcte_error_type apply_link_type (
    in Pcte_type_name_in_sds link_type,
    in Pcte_type_name_in_sds object_type
);
```

```
/* 10.2.4 SDS_CREATE_BOOLEAN_ATTRIBUTE_TYPE */
```

```
Pcte_error_type create_boolean_attribute_type (
    in Pcte_name local_name,
    in Pcte_boolean initial_value,
    in Pcte_duplication duplication,
    out Pcte_type_name_in_sds new_type
);
```

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

```
/* 10.2.5 SDS_CREATE_DESIGNATION_LINK_TYPE */
```

```
Pcte_error_type create_designation_link_type (
    in Pcte_name local_name,
    in Pcte_natural lower_bound,
    in Pcte_natural upper_bound,
    in Pcte_duplication duplication,
    in Pcte_key_types_in_sds key_types,
    out Pcte_type_name_in_sds new_type
);
```

```
/* The effect of not providing the optional parameter local_name to the abstract operation is */
/* achieved by specifying local_name as NULL. The effect of not providing the optional */
/* parameter upper_bound to the abstract operation is achieved by specifying upper_bound */
/* as 0. */
```

```
/* 10.2.6 SDS_CREATE_ENUMERAL_TYPE */
```

```
Pcte_error_type create_enumeral_type (
    in Pcte_name local_name,
    out Pcte_type_name_in_sds new_type
);
```

```
/* 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_ENUMERATION_ATTRIBUTE_TYPE */
```

```
Pcte_error_type create_enumeration_attribute_type (
    in Pcte_name          local_name,
    in Pcte_type_names_in_sds values,
    in Pcte_duplication    duplication,
    in Pcte_natural        initial_value,
    out Pcte_type_name_in_sds new_type
);
```

```
/* The effect of not providing the optional parameter local_name to the abstract operation is */
/* achieved by specifying local_name as NULL. The effect of not providing the optional */
/* parameter initial_value to the abstract operation is achieved by specifying initial_value */
/* as 0. */
```

```
/* 10.2.8 SDS_CREATE_FLOAT_ATTRIBUTE_TYPE */
```

```
Pcte_error_type create_float_attribute_type (
    in Pcte_name          local_name,
    in Pcte_float          initial_value,
    in Pcte_duplication    duplication,
    out Pcte_type_name_in_sds new_type
);
```

```
/* The effect of not providing the optional parameter local_name to the abstract operation is */
/* achieved by specifying local_name as NULL. The effect of not providing the optional */
/* parameter initial_value to the abstract operation is achieved by specifying initial_value */
/* as 0.0. */
```

```
/* 10.2.9 SDS_CREATE_INTEGER_ATTRIBUTE_TYPE */
```

```
Pcte_error_type create_integer_attribute_type (
    in Pcte_name          local_name,
    in Pcte_integer        initial_value,
    in Pcte_duplication    duplication,
    out Pcte_type_name_in_sds new_type
);
```

```
/* The effect of not providing the optional parameter local_name to the abstract operation is */
/* achieved by specifying local_name as NULL. The effect of not providing the optional */
/* parameter initial_value to the abstract operation is achieved by specifying initial_value */
/* as 0. */
```

```
/* 10.2.10 SDS_CREATE_NATURAL_ATTRIBUTE_TYPE */
```

```
Pcte_error_type create_natural_attribute_type (
    in Pcte_name          local_name,
    in Pcte_natural        initial_value,
    in Pcte_duplication    duplication,
    out Pcte_type_name_in_sds new_type
);
```

```

/* The effect of not providing the optional parameter local_name to the abstract operation is */
/* achieved by specifying local_name as NULL. The effect of not providing the optional */
/* parameter initial_value to the abstract operation is achieved by specifying initial_value */
/* as 0. */

```

```

/* 10.2.11 SDS_CREATE_OBJECT_TYPE */

```

```

Pcte_error_type create_object_type (
    in Pcte_name                local_name,
    in Pcte_type_names_in_sds    parents,
    out Pcte_type_name_in_sds    new_type
);

```

```

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

```

```

/* 10.2.12 SDS_CREATE_RELATIONSHIP_TYPE */

```

```

Pcte_error_type create_relationship_type (
    in Pcte_name                forward_local_name,
    in Pcte_link_type_properties forward_properties,
    in Pcte_key_types_in_sds    forward_key_types,
    in Pcte_name                reverse_local_name,
    in Pcte_link_type_properties reverse_properties,
    in Pcte_key_types_in_sds    reverse_key_types,
    out Pcte_type_name_in_sds    forward_type,
    out Pcte_type_name_in_sds    reverse_type
);

```

```

/* The effect of not providing the optional parameter forward_local_name to the abstract */
/* operation is achieved by specifying forward_local_name as NULL. The effect of not */
/* providing the optional parameter reverse_local_name to the abstract operation is achieved */
/* by specifying reverse_local_name as NULL. The effect of not providing the optional */
/* parameter forward_upper_bound to the abstract operation is achieved by specifying */
/* forward_properties.upper_bound as 0. The effect of not providing the optional */
/* parameter reverse_upper_bound to the abstract operation is achieved by specifying */
/* reverse_properties.upper_bound as 0. */

```

```

/* 10.2.13 SDS_CREATE_STRING_ATTRIBUTE_TYPE */

```

```

Pcte_error_type create_string_attribute_type (
    in Pcte_name                local_name,
    in Pcte_string              initial_value,
    in Pcte_duplication          duplication,
    out Pcte_type_name_in_sds    new_type
);

```

```

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

```

/* 10.2.14 SDS_CREATE_TIME_ATTRIBUTE_TYPE */

```
Pcte_error_type create_time_attribute_type (
    in Pcte_name          local_name,
    in Pcte_time          initial_value,
    in Pcte_duplication    duplication,
    out Pcte_type_name_in_sds new_type
);
```

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

/* 10.2.15 SDS_GET_NAME */

```
Pcte_error_type get_name (
    out Pcte_name name
);
```

/* 10.2.16 SDS_IMPORT_ATTRIBUTE_TYPE */

```
Pcte_error_type import_attribute_type (
    in Pcte_object_reference from_sds,
    in Pcte_type_name_in_sds type,
    in Pcte_name          local_name
);
```

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

/* 10.2.17 SDS_IMPORT_ENUMERAL_TYPE */

```
Pcte_error_type import_enumeral_type (
    in Pcte_object_reference from_sds,
    in Pcte_type_name_in_sds type,
    in Pcte_name          local_name
);
```

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

/* 10.2.18 SDS_IMPORT_LINK_TYPE */

```
Pcte_error_type import_link_type (
    in Pcte_object_reference from_sds,
    in Pcte_type_name_in_sds type,
    in Pcte_name          local_name
);
```

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

/* 10.2.19 SDS_IMPORT_OBJECT_TYPE */

```
Pcte_error_type import_object_type (  
    in Pcte_object_reference    from_sds,  
    in Pcte_type_name_in_sds    type,  
    in Pcte_name                local_name  
);
```

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

/* 10.2.20 SDS_INITIALIZE */

```
Pcte_error_type initialize (  
    in Pcte_name    name  
);
```

/* 10.2.21 SDS_REMOVE */

```
Pcte_error_type remove (  
);
```

/* 10.2.22 SDS_REMOVE_DESTINATION */

```
Pcte_error_type remove_destination (  
    in Pcte_type_name_in_sds    link_type,  
    in Pcte_type_name_in_sds    object_type  
);
```

/* 10.2.23 SDS_REMOVE_TYPE */

```
Pcte_error_type remove_type (  
    in Pcte_type_name_in_sds    type  
);
```

/* 10.2.24 SDS_SET_ENUMERAL_TYPE_IMAGE */

```
Pcte_error_type set_enumeral_type_image (  
    in Pcte_type_name_in_sds    type,  
    in Pcte_enumeral_type_image image  
);
```

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

/* 10.2.25 SDS_SET_TYPE_MODES */

```
Pcte_error_type set_usage_mode (  
    in Pcte_type_name_in_sds    type,  
    in Pcte_definition_mode_values    usage_mode  
);
```

```

Pcte_error_type set_export_mode (
    in Pcte_type_name_in_sds      type,
    in Pcte_definition_mode_values export_mode
);
/* The effect of not providing the optional parameter export_mode to the abstract operation */
/* is obtained by calling Pcte_sds_set_usage_mode. The effect of not providing the optional */
/* parameter usage_mode is obtained by calling Pcte_sds_set_export_mode. The effect of */
/* providing both optional parameters usage_mode and export_mode is obtained by calling */
/* Pcte_sds_set_usage_mode and Pcte_sds_set_export_mode in sequence. As an operation */
/* call with neither optional parameter has no effect, no means for making such a call is */
/* provided. */
/* 10.2.26 SDS_SET_TYPE_NAME */
Pcte_error_type set_type_name (
    in Pcte_type_name_in_sds type,
    in Pcte_name              local_name
);
/* The effect of not providing the optional parameter image to the abstract operation is */
/* achieved by specifying image as NULL. */
/* 10.2.27 SDS_UNAPPLY_ATTRIBUTE_TYPE */
Pcte_error_type unapply_attribute_type (
    in Pcte_type_name_in_sds attribute_type,
    in Pcte_type_name_in_sds type
);
/* 10.2.28 SDS_UNAPPLY_LINK_TYPE */
Pcte_error_type unapply_link_type (
    in Pcte_type_name_in_sds link_type,
    in Pcte_type_name_in_sds object_type
);

```

10.3 Usage operations

```

/* 10.3.1 SDS_GET_ATTRIBUTE_TYPE_PROPERTIES */
Pcte_error_type get_attribute_type_properties (
    in Pcte_type_name_in_sds      type,
    out Pcte_duplication           duplication,
    out Pcte_value_type            value_type,
    out Pcte_enumeration_value_type_in_sds enumeration_value_type,
    out Pcte_attribute_value       initial_value
);
/* If the abstract operation returns an enumeration value type in value_type then value_type */
/* is set to PCTE_ENUMERATION_VALUE_TYPE and enumeration_value_type */
/* contains the sequence of enumeration value type nominators. */

```

```

/* 10.3.2 SDS_GET_ENUMERAL_TYPE_IMAGE */
Pcte_error_type get_enumeral_type_image (
    in Pcte_type_name_in_sds      enumeral_type,
    out Pcte_enumeral_type_image  image
);

/* 10.3.3 SDS_GET_ENUMERAL_TYPE_POSITION */
Pcte_error_type get_enumeral_type_position (
    in Pcte_type_name_in_sds  enumeral_type,
    in Pcte_type_name_in_sds  attribute_type,
    out Pcte_natural          position
);

/* 10.3.4 SDS_GET_LINK_TYPE_PROPERTIES */
Pcte_error_type get_link_type_properties (
    in Pcte_type_name_in_sds    type,
    out Pcte_link_type_properties properties,
    out Pcte_key_types_in_sds    key_types,
    out Pcte_type_name_in_sds    reverse
);

/* The category, lower bound, upper bound, exclusiveness, stability, duplication, key types, */
/* and reverse values are returned in the members with the corresponding names of the */
/* Pcte_link_type_properties object pointed to by properties. If the abstract operation */
/* returns no value in reverse , reverse is set to NULL. */

/* 10.3.5 SDS_GET_OBJECT_TYPE_PROPERTIES */
Pcte_error_type get_object_type_properties (
    in Pcte_type_name_in_sds    type,
    out Pcte_contents_type      contents_type,
    out Pcte_type_names_in_sds  parents,
    out Pcte_type_names_in_sds  children
);

/* If the abstract operation returns no value in contents_type then contents_type is set to */
/* PCTE_NO_CONTENTS. */

/* 10.3.6 SDS_GET_TYPE_KIND */
Pcte_error_type get_type_kind (
    in Pcte_type_name_in_sds  type,
    out Pcte_type_kind        type_kind
);

```

```
/* 10.3.7 SDS_GET_TYPE_MODES */
```

```
Pcte_error_type get_type_modes (  
    in Pcte_type_name_in_sds      type,  
    out Pcte_definition_mode_values usage_mode,  
    out Pcte_definition_mode_values export_mode,  
    out Pcte_definition_mode_values max_usage_mode  
);
```

```
/* 10.3.8 SDS_GET_TYPE_NAME */
```

```
Pcte_error_type get_type_name (  
    in Pcte_type_name_in_sds type,  
    out Pcte_type_name      name  
);
```

```
/* 10.3.9 SDS_SCAN_ATTRIBUTE_TYPE */
```

```
Pcte_error_type scan_attribute_type (  
    in Pcte_type_name_in_sds      type,  
    in Pcte_attribute_scan_kind    scanning_kind,  
    out Pcte_type_names_in_sds    types  
);
```

```
/* 10.3.10 SDS_SCAN_ENUMERAL_TYPE */
```

```
Pcte_error_type scan_enumeral_type (  
    in Pcte_type_name_in_sds      type,  
    out Pcte_type_names_in_sds    types  
);
```

```
/* 10.3.11 SDS_SCAN_LINK_TYPE */
```

```
Pcte_error_type scan_link_type (  
    in Pcte_type_name_in_sds      type,  
    in Pcte_link_scan_kind        scanning_kind,  
    out Pcte_type_names_in_sds    types  
);
```

```
/* 10.3.12 SDS_SCAN_OBJECT_TYPE */
```

```
Pcte_error_type scan_object_type (  
    in Pcte_type_name_in_sds      type,  
    in Pcte_object_scan_kind      scanning_kind,  
    out Pcte_type_names_in_sds    types  
);
```

```
/* 10.3.13 SDS_SCAN_TYPES */
```

```
Pcte_error_type scan_types (  
    in Pcte_type_kind            kind,  
    out Pcte_type_names_in_sds   types  
);
```