Standard ECMA-270
December 1997

ECMA

Standardizing Information and Communication Systems

Portable Common Tool Environment (PCTE) - Mapping from CASE Data Interchange Format (CDIF) to PCTE

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

Standard ECMA-270
December 1997

ECMA

Standardizing Information and Communication Systems

Portable Common Tool Environment (PCTE) - Mapping from CASE Data Interchange Format (CDIF) to PCTE

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

IW Ecma-270.doc 07-01-98 15,45

Brief History

PCTE, Portable Common Tool Environment, is an interface standard. The interface is designed to support program portability by providing machine-independent access to a set of facilities. These facilities, which are described in the PCTE Abstract Specification (Standard ECMA-149), are designed particularly to provide an infrastructure for programs which may be part of environments supporting systems engineering projects. Such programs, which are used as aids to systems development, are often referred to as tools.

CDIF, CASE Data Interchange Format, is a standard of the Electronic Industries Association. CDIF was defined primarily as a standard for exchanging models between CASE tools. Since it is necessary for parties that exchange models to have a common understanding of them, CDIF is not just a standard for a transfer format, but also for an integrated meta-model (schema) of the data and process models that can be exchanged. This harmonises the concepts of different methods and viewpoints, making CDIF independent of particular methods and tools.

This ECMA Standard for a mapping from CDIF to PCTE allows PCTE schemas (SDSs) to be derived from the CDIF integrated meta-model. Models defined with such SDSs can be freely exchanged using CDIF. Derived SDSs are included in this ECMA Standard; they provide a substantial information model for any PCTE-based software engineering environment or repository.

From the first meeting of ECMA/TC33 for standardizing PCTE, it was recognized that additional standards would be required to achieve the final aim of integrating independently produced tools in a software engineering environment. A task group (Task Group for the Reference Model, TGRM) was formed to study the requirements. Two of the areas in which further standards were required were data interchange between PCTE-based repositories and SDSs for software engineering tools.

After ECMA/TC33 received a presentation of CDIF in 1991, there was enough interest in the possibility of basing a PCTE standard for data exchange on the use of CDIF for TC33 to extend the scope of TGRM for such work. TGRM was later renamed Task Group for Data Interchange, TGDI. The work on data exchange progressed with the attendance of CDIF representatives. This close co-operation was formalised by the agreement of a Memorandum of Understanding between ECMA/TC33 and the EIA CDIF Division.

Meanwhile an international initiative had started in 1991 for co-ordinating the activities of several bodies concerned with standards for describing or interchanging software engineering models. This resulted in the approval of an ISO/IEC JTC1 project 7.28 Software Engineering Data Description and Interchange (SEDDI) which was assigned to SC7/WG11 in 1992. JTC1/SC7 accepted EIA CDIF as a Category C liaison. New versions of CDIF, taking account of WG11 comments on internationalization, were published as EIA Interim Standards from 1994 onwards. WG11 agreed to use these versions for its SEDDI standard.

The scope of SEDDI included PCTE SDSs corresponding to the SEDDI meta-model. The PCTE SDSs were to be produced in conjunction with ECMA/TC33 through ECMA's Category A liaison with JTC1. The work progressed slowly in ECMA/TC33 owing to its limited resources being focused on the extensions of PCTE for fine-grain objects and object-orientation. It was given a fresh impetus in 1996 with a joint project between Fujitsu and ICL for prototyping CDIF with a PCTE repository. The work was completed by ECMA/TC33 in close liaison with JTC1/SC7/WG11 and JTC1/SC22/WG22 (PCTE).

This ECMA Standard has been adopted by the General Assembly of December 1997.

Table of contents

1 Scope

This ECMA Standard specifies a mapping of CDIF to PCTE or, more specifically, of a CDIF subject area to a PCTE Schema Definition Set (SDS).

The mapping may be applied to standard CDIF subject areas and to CDIF subject areas defined as user extensions, provided they follow the restrictions used in the CDIF subject area standards.

PCTE (ECMA-149) 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* or *CASE tools*.

CDIF (EIA/IS-107) is an architecture and facilities for transferring information, often referred to as *models*, between CASE tools, including repositories. The CDIF architecture includes a modelling language, the CDIF meta-meta-model, for defining an integrated meta-model that in turn defines the information models for subject areas of systems engineering. The CDIF architecture also includes a transfer format for transferring models between tools.

Application of this mapping to a CDIF subject area generates a derived PCTE SDS that is semantically equivalent to the CDIF subject area. Such *derived SDSs* provide

- a means of exchanging models between CASE tools and a PCTE implementation;
- a means of realising models defined according to the corresponding CDIF subject areas in a PCTE installation;
- hence a basis for standard SDSs for systems engineering subject areas.

The derived SDSs are not sufficient

- to define all the properties needed for efficient use of the models within a PCTE installation;
- for the faithful transfer of the models between different PCTE installations.

2 Conformance

2.1 Conformance of an SDS

A PCTE SDS conforms to this ECMA Standard with respect to a given CDIF subject area if, and only if, it contains type definitions that are derived from the specification for the given CDIF subject area (as defined in the relevant EIA/IS subject area specifications) according to the mapping in clause 9. Such an SDS is referred to as an *SDS for the (given) CDIF subject area*.

2.2 Conformance of a PCTE tool

A PCTE model (of a given CDIF subject area) is a collection of objects which are instances of types defined in an SDS, which conforms to this ECMA Standard, for the given subject area. This ECMA Standard does not define how a PCTE model is represented in a PCTE installation.

NOTE

A PCTE model could, for example, be represented by a composite object whose components are instances of the types defined in a conforming SDS or by a directory object with existence links to instances of the types defined in a conforming SDS.

A PCTE tool conforms to this ECMA Standard if and only if

- it uses an SDS for a given CDIF subject area to manage a PCTE model of that subject area, and
- it uses no alternative SDS to model properties of that subject area

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this ECMA Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this ECMA Standard are encouraged to investigate the possibility

of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

ECMA-149	Portable Common Tool Environment (PCTE) - Abstract Specification, 4th edition (1997)
EIA/IS-107	CDIF / Framework for Modeling and Extensibility, January 1994
EIA/IS-111	CDIF - Integrated Meta-model / Foundation Subject Area, January 1994
EIA/IS-112	CDIF - Integrated Meta-model / Common Subject Area, December 1995
EIA/IS-113	CDIF - Integrated Meta-model / Data Definition Subject Area, May 1996 (draft)
EIA/IS-114	CDIF - Integrated Meta-model / Data Modeling Subject Area, December 1996

4 Terms and definitions

4.1 CDIF and PCTE terms

Most of the technical terms used in this ECMA Standard are CDIF and PCTE terms defined in EIA/IS-107 and ECMA-149, respectively. Such terms are qualified by "CDIF" and "PCTE", respectively, as necessary to make the intended meaning clear.

There are two areas of difficulty with terminology that are inherent in the subject matter. The first is that terms such as "subject area" and "meta-object" in CDIF, "SDS" and "object type" in PCTE, are used either in a general, inclusive sense that captures the informal notion, e.g. a subject area with all its contents, the type of an object including its attributes and links, or in a restricted, exclusive sense that is required for technical specification, where for example a subject area is separate from the meta-entities etc. that are *used* in it, an object type is separate from the attribute types etc. that may be *applied* to it. In EIA/IS-107, such terms are used in the restricted sense, except occasionally in the more general discussion (clauses 7 and 8) where the distinction is clear from the context.

The second area of difficulty is groups of closely related concepts with conventionally related ways of referring to them. For example, in CDIF there are (a) meta-entity, (b) the meta-meta-entity "MetaEntity", and (c) metaentity (an instance of "MetaEntity"); in PCTE there are (a) object type in DDL, (b) object type (and type in SDS) in the PCTE Foundation, and (c) object type (and type in SDS) in the SDS "metasds". For EIA/IS-107, the simplest and most readable form has been chosen, *viz* terms used in the textual specification of subject areas (e.g. meta-entity) and SDSs (e.g. object type).

4.2 Other terms

For the purposes of this ECMA Standard, the terms and definitions given in ECMA-149, EIA/IS-107, EIA/IS-111, EIA/IS-112, EIA/IS-113, EIA/IS-114 and the following apply.

4.2.1 CASE tool

A program that is used as an aid to systems development.

4.2.2 derived SDS

A PCTE SDS generated by application of the mapping defined in this ECMA Standard to a CDIF subject area.

4.2.3 model

(When used generally) a description of an information system used in system development.

4.2.4 PCTE model

See 2.2.

4.2.5 PCTE tool

A CASE tool that uses the facilities of a PCTE implementation.

4.2.6 SDS for a (given) CDIF subject area

See 2.1.

5 Symbols (and abbreviated terms)

5.1 Notations

PCTE SDSs are defined using PCTE DDL as defined in annex B of ECMA-149.

5.2 Abbreviations

The following abbreviations are used in this ECMA Standard.

5.2.1 CDIF

CASE Data Interchange Format.

5.2.2 DDL

Data Definition Language.

5.2.3 PCTE

Portable Common Tool Environment.

5.2.4 SDS

Schema Definition Set (a PCTE term).

6 Outline of the Standard

Clause 7 states the general principles that are applied to define the mapping. Clause 8 discusses the main differences between CDIF and PCTE and how these are resolved in the mapping. The mapping is defined in clause 9. Clause 10 introduces some examples of derived SDSs which follow in informative annexes.

7 Mapping principles

This clause defines general principles for deriving PCTE SDSs corresponding to subject areas of the CDIF integrated meta-model.

The following general principles were applied when defining the mapping of CDIF to PCTE.

a) The purpose of the mapping is to derive PCTE SDSs corresponding to CDIF subject areas that enable the realisation in a PCTE installation of models defined according to those subject areas.

b) The derived SDSs are not intended to be sufficient for other purposes such as the definition of all the properties needed for efficient use of the models within a PCTE installation nor for the faithful transfer of the models between different PCTE installations. Thus, for example, the derived SDSs make no use of object contents or most link categories.

c) The mapping should be capable of deriving valid SDSs for subject areas defined as user extensions, provided they follow the restrictions used in the CDIF subject area standards. The mapping cannot resolve any clashes in user extensions that are defined independently and then used together.

d) The mapping should be fully determined by rules that can be applied automatically, either by human or by computer.

e) The mapping should be a total function from CDIF to PCTE, i.e. a mapping of all CDIF concepts to a subset of PCTE concepts. With the mapping represented as a two-column table, the CDIF column should contain all CDIF concepts but the PCTE column may omit PCTE concepts that are not used to represent CDIF concepts.

f) The mapping should not preclude the addition of PCTE-implementation-dependent DDL to derived SDSs that may be required by a conforming PCTE implementation. For example, the mapping might allow additional importations that are not defined by EIA/IS-107.

g) The mapping should derive a single, unique PCTE SDS from each CDIF subject area, so that a PCTE tool using information defined in a set of CDIF subject areas should only need the corresponding PCTE SDSs in its working schema. (A possible exception might be for one SDS that is needed for use with any subject area, but this has not proved necessary.)

h) The semantics of a derived SDS should, whenever possible, be the same as the semantics of the corresponding subject area, and should be determined by reference to the standard for that subject area.

i) The derived PCTE SDSs should have names that distinguish them from SDSs for similar "subject areas" that are not derived from CDIF subject areas.

j) The names of the derived SDSs should allow the management of different versions within a PCTE installation. For example the mapping might allow the SDS name to have a suffix not defined in this Standard.

k) Names should be retained from the CDIF subject area standard as far as possible. Any additional names should be chosen appropriately for their meanings.

8 Issues for the mapping

8.1 Mapping of names

8.1.1 Uniqueness of Names

In CDIF, uniqueness of names is defined within the scope of a working meta-model (a set of subject areas): subject area names, meta-entity names, full meta-relationship names and full meta-attribute names must be unique, where the full meta-relationship name includes the concatenated names of the source and destination meta-entities and the full meta-attribute name includes the concatenated name of the containing meta-entity or meta-relationship. In PCTE, uniqueness of names is defined within the scope of an SDS and resolution of name clashes is defined for a working schema (an ordered set of SDSs).

CDIF does not allow the definition of local names that are limited to the scope of a subject area. PCTE allows the definition of local names that are limited to the scope of an SDS.

The CDIF rules for uniqueness of full names are more restrictive than the PCTE rules, but the CDIF rules allow duplication of (simple) names of meta-attributes and meta-relationships within a subject area.

8.1.2 Syntax of Names

In CDIF, the maximum length of a meta-object name is 32 (within CDIF standards) (see EIA/IS-107). In PCTE, the maximum length of an SDS name or a local name for a type within an SDS is an implementation limit MAX_NAME_SIZE which must be at least 31; this limit of 31 is used within ECMA-149.

Otherwise, CDIF and PCTE follow identical rules for the syntax of names, including case insensitivity, except that CDIF allows hyphen "-" to be included and names to start with a digit. Hyphen is used in PCTE to construct a type name that is unique across an installation from an SDS name and a type name within the SDS.

The conventions for constructing meaningful names in the standards differ: in CDIF names are concatenated from words starting with capital letters, in PCTE names are concatenated from words, without initial capitals, separated by underscore "_".

PCTE names are derived from CDIF names with minimum change as defined in 9.2.

8.1.3 Basic mapping of names

CDIF meta-relationships or meta-attributes with the same simple name are intended to have the same semantics (in both denotation and connotation). If each meta-relationship (or meta-attribute) were mapped to a separate object type (attribute type) this similarity would be lost, so the PCTE names are derived from the simple CDIF names. The consequent problems of duplication are discussed below (see 8.1.4).

8.1.4 Duplication of names

The simple names of meta-attributes may be duplicated in different meta-entities and meta-relationships. Such duplication can be considered as reuse or implicit specialisation: for example, in the subject area Common there are three meta-attributes called Name, each being specialised to be the name of the containing meta-entity. This specialisation mainly affects the semantics, the only exception in this example being that two of the Name meta-attributes have data type String, while one, contained by AbstractionLevel, has data type Enumerated (which may be treated as a specialisation of String). Any exception is a CDIF modelling error.

The names of meta-relationships may be duplicated in meta-relationships whose full names have different source or destination meta-entities. A particular case of this occurs with inheritance of meta-relationships, where the subtype usually has the same (simple) name as a supertype. Duplication of meta-relationship names can be treated analogously to duplication of meta-attribute names. Such duplication can be considered as reuse or implicit specialisation: for example, the subject area DataModelling has several meta-relationships Incorporates,

each specialising the notion of the source meta-entity incorporating the destination meta-entity. Any exception is a CDIF modelling error.

8.2 Subtyping of meta-entities

CDIF and PCTE support subtyping of meta-entities and object types, respectively, from one or more supertypes, with the same static rules but for the following exception. As a consequence of CDIF's approach to the uniqueness of names, a meta-entity may inherit meta-attributes of the same name from different direct supertypes. If these supertypes have inherited these meta-attributes, directly or indirectly, from a common supertype, the meta-attribute is inherited (once); otherwise, it is a CDIF modelling error. This modelling error cannot occur in PCTE: attribute type names are unique within an SDS, so two object types cannot have different attribute types of the same name applied to them.

The CDIF meta-entity hierarchy maps to the PCTE object hierarchy indirectly through the direct mapping of each CDIF supertype reference to a PCTE parent type reference to the object type derived from the supertype.

8.3 Subtyping of meta-relationships

CDIF supports subtyping of meta-relationships. Although PCTE does not support subtyping of link types, the ability to define a link type separately from any object type, to apply it to more than one source object and to extend it to more than one destination object type provides similar semantics.

CDIF inheritance is little used for properties which could be mapped to PCTE concepts that are represented syntactically in PCTE DDL. The only meta-attributes that are inherited are

a) those (inherited from the root of the meta-relationship hierarchy RootEntity.IsRelatedTo.RootEntity) that provide common meta-attributes (CDIFIdentifier, DateCreated, DateUpdated, TimeCreated, TimeUpdated) for all meta-relationships; these properties are not relevant to the usage of PCTE link types;

b) SequenceNumber that provides a key for some meta-relationships whose maximum destination cardinality is N; this property is provided, in all cases, for PCTE usage by the key that is required for any link type of cardinality many.

These CDIF properties need not be mapped directly to PCTE concepts.

In most cases of inheritance the simple meta-relationship name, i.e. the name without the source and destination meta-entities, is not changed and inheritance is treated as a case of duplication of names.

8.4 Modularity and sharing

CDIF and PCTE both support the concept of a dynamically changing schema which can be defined in modules (subject areas and SDSs, respectively) that may contain shared and extended definitions.

In CDIF, a *collectable meta-object* is *used* in one or more subject areas with the following rules (see EIA/IS-107)

a) A meta-entity may have different local meta-attributes and meta-relationships in different subject areas.

b) A meta-relationship may have different local meta-attributes in different subject areas.

c) A meta-entity or meta-relationship may be used in a subject area without a supertype being used explicitly, but its inherited meta-entities and meta-relationships are also used implicitly in that subject area.

d) A meta-relationship may only be used in a subject area if the source and destination meta-entities are also used in that subject area.

e) A meta-attribute may only be used in a subject area if the meta-entity or meta-relationship that it describes is also used in that subject area.

In PCTE, a *type* is *declared* in one SDS and may be *imported* into one or more other SDSs with the following rules (see ECMA-149)

a) An object type may be defined or imported in an SDS without having any attribute type or link type applied explicitly in that SDS.

b) When an object type is imported to an SDS, its ancestor types are imported implicitly to that SDS.

c) When an object type is imported to an SDS, any applied attribute types and link types are not imported implicitly to that SDS, nor is their application to the object type imported implicitly for applied attribute types and link types that are imported explicitly to that SDS.

d) A link type may be defined or imported to an SDS without having any attribute type applied explicitly in that SDS.

e) When a link type is imported to an SDS, any key attribute types and its reverse link type with any key attribute types are imported implicitly to that SDS and key attribute types are applied.

f) When a link type is imported to an SDS, any applied non-key attribute types and link types are not imported implicitly to that SDS, nor is their application to the link type imported implicitly for applied attribute types that are imported explicitly to that SDS.

g) An attribute type may be defined or imported to an SDS without being applied explicitly to any object type or link type in that SDS.

h) When an attribute type is imported to an SDS, it is not applied implicitly to any object type or link type in that SDS.

i) When an enumeration attribute type is imported to an SDS, its enumeration types are imported implicitly to that SDS.

These two approaches are quite close but a problem arises from differences in the scoping of names (see 8.1.1).

a) In CDIF, collectable meta-objects of the same type (meta-entity, meta-relationship, meta-attribute) with the same name in the same or different subject areas are the same meta-object.

b) In PCTE, types (object, link or attribute) with the same name in different SDSs are different types. A type may only occur in more than one SDS by importing the definition. (Types in the same SDS must have different names.)

The differences in scoping of names could be handled in two ways as follows.

a) The semantics, that meta-objects of the same name in different subject areas are the same meta-objects, can be captured by importing the derived type from one SDS into the SDSs of the other given subject areas and importing and applying any applied link types and attribute types derived from those subject areas. It would still be necessary, theoretically, to import and apply any inherited link types and attribute types, but failure to do this would lose little information in practice.

b) Meta-objects of the same name in different subject areas are treated as different and are mapped independently to types which are different. This loses the intention that such meta-objects are views of the same meta-object, but matches the way that CDIF allows the meta-objects to have different meta-relationships and meta-attributes in different subject areas.

The second way is used in this mapping. This approach conforms more closely to the principle that the mapping should derive a single, unique PCTE SDS from each CDIF subject area, so that a PCTE tool using information defined in a set of CDIF subject areas should only need the corresponding PCTE SDSs in its working schema. It also allows the use of a working schema including all the derived SDSs provided none of the derived types differs in different SDSs (although the application of types may vary).

8.5 Subject area Foundation

The subject area Foundation specifies those basic concepts that are required for any CDIF transfer. Hence it is required in any transfer. It is also a CDIF requirement that it can be used without any other subject area, to transfer entities, relationships and attributes as "raw data" without any additional attached meaning. Thus the subject area Foundation supplies no meaning for the integrated meta-model other than for the mechanics of transferring data.

9 Mapping a CDIF subject area to a PCTE SDS

9.1 Format of mapping

The fundamental concepts of CDIF and PCTE are defined in the CDIF meta-meta-model and the PCTE Foundation respectively. For the purpose of the mapping — deriving a PCTE SDS from a CDIF subject area — it is not necessary to define a mapping for all elements of the CDIF meta-meta-model, but only for those needed for the textual definition of a CDIF subject area. For example, MetaEntity in the meta-meta-model is needed but CollectableMetaObject is not.

The mapping is defined pragmatically and structured according to the format of a subject area definition. The mappings of major concepts — subject area, meta-entity, meta-relationship, meta-attribute, all of which are meta-objects — are defined in terms of the mappings of their properties (i.e. their meta-meta-attributes or meta-meta-relationships in the meta-meta-model) and these properties are ordered as in the definitions of instances of that meta-object in subject area definitions.

The format of each clause is as follows:

a) a heading with the name of a CDIF meta-object

b) a brief general statement of the mapping of the CDIF meta-object to a PCTE type

c) a two-column table defining the detailed mapping of the CDIF meta-object to the PCTE type.

The first row of the table gives the name of the CDIF meta-object and of the derived PCTE type. Each other row of the table gives the mapping of a property of the CDIF meta-object to PCTE concepts. The table includes conventional entries whose meaning is as follows.

–	Nothing of the property definition is mapped to PCTE concepts that can be represented syntactically in PCTE DDL; a specific comment or one of the following generic comments may be given in parentheses.
– (semantics)	The property may define semantics textually; the same semantics apply to the PCTE type derived from the meta-object.
– (documentary)	The property does not define any semantics; it does not affect the definition of the PCTE type derived from the meta-object.

NOTE

The value of a documentary property may nevertheless be represented for PCTE use, for example as a comment in a DDL definition of the derived SDS, as the value of a specially defined attribute of the object type CDIF-_Foundation-RootEntity, or as the value of a specially defined attribute of the "type_in_sds" object type.

Mappings of properties which are the same for all the types of meta-object are extracted into a preceding clause. The mappings for the meta-objects are followed by the mappings of data types and the subject area Foundation.

9.2 Common meta-object properties

All types of meta-object have a Name, which is mapped to a PCTE type name based on a *converted CDIF name* as indicated in the mapping for each type of meta-object. A converted CDIF name is a CDIF name, except that any hyphen "-" in the CDIF name is replaced by underscore "_".

NOTE

If the converted CDIF name is not a valid PCTE name, i.e. is more than 31 characters or starts with a digit, it is further converted, as an exception, e.g. by abbreviation or omission of parts of the name, or by reordering the name.

The following properties are common to all types of meta-object and are mapped in the same way for each type of meta-object.

Table 1 - Common meta-object properties

Property	Mapping
CDIFMetaIdentifier	– (documentary)
Description	– (semantics)
Usage	– (semantics)
Aliases	– (documentary)
Constraints	– (semantics)

9.3 Subject area

The CDIF subject area meta-object is mapped to a PCTE SDS with the same name preceded by "CDIF_" and optionally succeeded by a version identification. Only the name is mapped to PCTE concepts.

NOTE

A version identification might identify the version of any of the following — CDIF subject area, mapping or derived SDS.

Table 2 - Subject area to SDS

Subject area	Schema Definition Set (SDS)
Name	Name of the SDS: the converted CDIF name preceded by "CDIF_", optionally succeeded by "_" and additional characters that identify the version of the SDS. Thus "CDIF_" becomes a reserved prefix for PCTE SDSs.
VersionNumber	– (documentary)

9.4 Meta-entities

Each CDIF meta-entity in the CDIF subject area is mapped to a PCTE object type. Only the name, supertypes (i.e. SubtypeOf), local meta-attributes and local meta-relationships are mapped to PCTE concepts.

The CDIF common root RootObject is mapped to a PCTE object type which is used as the root of the hierarchy of derived object types. This object type is derived by importing "system-object", the root of the PCTE object hierarchy as RootObject.

Table 3 - Meta-entity to Object type

Meta-entity	Object type
Name	Name of the object type: the converted CDIF name
SubtypeOf	Each object type corresponding to a CDIF supertype is a parent type of this object type. "system-object" is imported as RootEntity for the root of the object type hierarchy
SupertypeOf	– (redundant: implied by SubtypeOf in the CDIF supertype)
Type	– (semantics)
Inherited meta-attribute	– (implicit in PCTE inheritance from supertype)
Local meta-attribute. See 9.6 for the full mapping of each meta-attribute	An attribute type which is applied to the object type.
Inherited meta-relationship	– (implicit in PCTE inheritance from supertype)
Local meta-relationship. See 9.5 for the full mapping of each meta-relationship	A link type which is applied to the object type.

NOTE

The derivation of RootObject and RootEntity, for which special rules apply, is described in 9.8.

9.5 Meta-relationships

Each CDIF meta-relationship in the CDIF subject area is mapped to a PCTE reference link type with an implicit reverse link type. The source and destination of the link type are the object types derived from the source and destination, respectively, of the meta-relationship. If the CDIF MaxDestCard is N, the link type has cardinality many and its key is "system-number". Only the name, destination cardinalities and local meta-attributes are mapped to PCTE concepts.

Meta-relationships with the same simple name (see 8.1.3) are mapped to a link type, referred to as a *common link type*, with properties that correspond to the most general properties of the CDIF meta-attributes, e.g. having the least constrained cardinalities. The link type has all attributes applied that are derived from meta-attributes of the meta-relationships with the same name. The link type is applied to all source object types derived from source meta-entities of each meta-relationship with the same name; the link type is extended to all destination object types derived from destination meta-entities of each meta-relationship with the same name. An application of the derived link type may have some semantics, e.g. applied attribute types, which is not valid for the corresponding meta-relationship.

Table 4 - Meta-relationship to Link type

Meta-relationship	Link type
Name	Name of the link type: the converted CDIF simple name of the meta-relationship.
SubtypeOf	– (information lost theoretically – see 8.3)
SupertypeOf	– (redundant: implied by SubtypeOf in the CDIF supertype)
MinSourceCard	– (semantics)
MaxSourceCard	– (semantics)
MinDestCard	The lower bound of the link type's cardinality
MaxDestCard	The upper bound of the link type's cardinality
Inherited meta-attribute	– (information lost theoretically – see 8.3)
Local meta-attribute. See 9.6 for the full mapping of each meta-attribute	An attribute type which is applied to the object type.

9.6 Meta-attributes

Each CDIF meta-attribute of a meta-entity or meta-relationship in the CDIF subject area is mapped to a PCTE attribute type. Only the name, supertypes (i.e. SubtypeOf), local meta-attributes and local meta-relationships are mapped.

Meta-attributes with the same simple name (see 8.1.3) are mapped to an attribute type, referred to as a *common attribute type*, with the PCTE value type that corresponds to the most general data type of those CDIF meta-attributes. An application of the derived attribute type may have some semantics, e.g. extra enumeration images, which is not valid for the corresponding meta-relationship.

Table 5 - Meta-attribute to Attribute type

Meta-attribute	Attribute type
Name	Name of the attribute type: the converted CDIF simple name of the meta-attribute.
DataType. See 9.7 for the full mapping of the data type	A value type for the attribute type.
Domain	– (semantics), except if the CDIF data type is Enumerated or Integer (see table 6)
Length	– (semantics)
IsOptional	– (semantics, with some information loss since each instance of a visible PCTE attribute type always has a value, possibly the default value)

9.7 Data types

Each CDIF data type defined in EIA/IS-107 is mapped to a PCTE value type, with the exception of the following CDIF data types which have no close PCTE equivalent and are not used in the CDIF integrated meta-model – Bitmap, IntegerList, Point, PointList, Date (for relative dates), Time (for relative times).

NOTE

The unrepresented data types, and unrepresented ranges of values, could be represented by a PCTE string which is coded to represent both the data type and its value.

The permitted ranges of values are not defined precisely for some CDIF data types or for all PCTE values types. The range of values for some CDIF data types are only defined in the transfer format syntax and might differ in different transfer formats. The ranges of values for other CDIF data types are specified in EIA/IS-107. The range of values for PCTE value types is implementation-defined under certain constraints on maximum and minimum values specified as implementation limits in ECMA-149. It is a general requirement of standards and implementations for both CDIF and PCTE that there should be no practical limitations on values: consequently the limits are likely to be extended over time to remove any limitations. Differences in permitted ranges of values for CDIF and PCTE have no practical consequences for the purpose of the mapping — deriving a PCTE SDS from a CDIF subject area.

Table 6 - Data type to Value type

Data type	Value type
Bitmap	– (a coded string could be used)
Boolean	boolean
Date (absolute)	time, with the "time" set to "T00:00:00Z"
Date (relative)	– (a coded string could be used)
Enumerated	enumeration, where the sequence of enumeration images is derived from the sequence of values of the CDIF Domain of the meta-attribute (see table 5) or, if the enumeration is the value type of a common attribute type (see 9.6), the union of the sequences of CDIF Domain values of the meta-attributes with the same name
Float	float
Identifier	string
Integer (Domain is Positive integer)	natural
Integer (Domain is not Positive integer)	integer
IntegerList	– (a coded string could be used)
Point	– (a coded string could be used)
PointList	– (a coded string could be used)
String	string
Text	string
Time (absolute)	time, with the "date" set to "1980-01-01"
Time (relative)	– (a coded string could be used)

9.8 Subject area Foundation

The mapping of Foundation has special cases which are shown in the following table.

Table 7 - Subject area Foundation

Concept	Reason
RootObject — the common root of the CDIF meta-entity and meta-relationship hierarchies	Since PCTE has no inheritance for link types, there is no need to map the common root
RootEntity — the root of the CDIF meta-entity hierarchy	"system-object" is imported as RootEntity for the root of the object type hierarchy
RootEntity.IsRelatedTo.RootEntity — the root of the CDIF meta-relationship hierarchy	Since PCTE has no inheritance for link types, there is no need to map the root of the CDIF meta-relationship hierarchy
CDIFIdentifier, DateCreated, DateUpdated, TimeCreated, TimeUpdated — meta-attributes of RootObject and hence all meta-entities and meta-relationships	PCTE object type "system-object" has attributes — exact_identifier, last_access_time, last_modification_time, and last_change_time — with similar functions. A link is part of an object and hence has little need for separate values. If required, attributes derived by the rules in 9.6, can be applied to the object type RootEntity

10 Introduction to derived SDSs defined in annexes

10.1 Aims

Annexes A to D contain SDSs derived using the mapping of clause 9. This clause describes how the SDS definitions are structured in order to meet the conditions that

a) they will be compiled correctly by any compiler that conforms to ECMA-149, and

b) the mapping is shown as transparently as possible.

10.2 PCTE modelling language standard

SDSs derived from the mapping conform to the PCTE Data Definition Language (DDL) defined in ECMA-149.

10.3 Ordering of subject area definitions

Subject area definitions in CDIF standards follow an alphabetical ordering within a broad logical framework, described in EIA/IS-107. Slightly simplified the sequence is as follows.

a) The hierarchies of meta-entities and meta-relationships, as a list of names, with inheritance shown by relative indentation. This part of the definition is redundant since it is implied by information in the subsequent parts, but it provides a useful summary of the broad structure of the subject area.

b) Definitions of meta-entities, including the names of their meta-attributes and meta-relationships, inherited and local (i.e. defined specifically for the meta-entity), and definitions of the local meta-attributes.

c) Definitions of meta-relationships, including the names of their meta-attributes, inherited and local, and definitions of the local meta-attributes.

10.4 Ordering of SDS definitions

SDS definitions in ECMA-149 follow an ordering constrained by PCTE DDL syntax. In PCTE DDL, type names must be declared before use, except for destination object types and reverse link types in link type declarations. The

alphabetical ordering used in CDIF subject area definitions is maintained by first defining each object type without any applied attribute types or link types; the object type definitions are later extended in alphabetical order.

10.5 Structure of SDS definitions

The SDS definitions in annexes A to D are ordered as follows.

a) The derived SDS name in its simplest form, i.e. with no version identification.

b) A comment defining the version of the subject area from which the SDS is derived.

c) Import statements for attributes, defined in the predefined PCTE SDSs in ECMA-149, that are required in most or all user-defined SDSs (perhaps only for particular PCTE implementations). These imported attributes are not derived from the subject area.

d) The hierarchy of object types, derived from the CDIF meta-entity hierarchy, as a list of either simple object type declarations without any applied link or attribute types or, for object types derived from meta-entities defined in another subject area, import statements. The hierarchy is indicated by the names of the parent types in the **child type of** construction. It may additionally be shown by relative indentation as in CDIF, but this is less satisfactory with a deep hierarchy; the annexes vary in their approach.

e) Declarations of common attribute types, i.e. attribute types derived from CDIF meta-attributes with names duplicated in the current or a preceding subject area. In attribute type declarations and applications (in this and following clauses) the original CDIF data type is given in a comment where

   1) the CDIF data type is not String but maps to PCTE string,

   2) the application of a common attribute type has a PCTE value type which is an exception to the mapping rules for CDIF data types (see 9.7), and

   3) the CDIF data type is Enumerated and the derived type of the meta-attribute is the application of a common attribute type whose enumeration type has a larger range of enumeration images.

f) Declarations of link types derived from the definitions of meta-relationships, including declarations (or, for common attribute types, applications) of attribute types derived from the definitions of any local meta-attributes. Where the same link type is derived from more than one meta-relationship, the declaration becomes an extension to a further destination object for each such meta-relationship after the first. Applications of the link type with semantics which are not valid for the corresponding meta-relationship (see 9.5) are not indicated by DDL comments.

g) Extensions of object type definitions derived from the definitions of meta-entities, including declarations (or, for common attribute types, applications) of link types and attribute types derived from the definitions of any local meta-relationships and local meta-attributes respectively. Where there is no need for an extension, this is indicated by a comment "-- No extension".

h) The **end** construction for the SDS.

The SDS definitions contain comments starting "-- CDIF: " to indicate the CDIF concept from which the following DDL is derived.

Annex A

(informative)

SDS CDIF_Foundation

```
sds CDIF_ Foundation:

-- This annex defines the PCTE SDS CDIF_ Foundation that corresponds to
--    the CDIF subject area Foundation version 01.00 defined in
--    EIA/IS-111, CDIF - Integrated Meta-model / Foundation Subject Area, January 1994

import object type system-object as RootEntity;                    -- root of hierarchy for CDIF object types

end CDIF_Foundation;
```

Annex B

(informative)

SDS CDIF_Common

B.1 Introductory elements

sds CDIF_Common:

-- This annex defines the PCTE SDS CDIF_Common that corresponds to
-- the CDIF subject area Common version 01.00 defined in
-- EIA/IS-112, CDIF - Integrated Meta-model / Common Subject Area, December 1995

import attribute type system-number; -- required for link keys
import attribute type system-system_key; -- may be required for creating implicit reverse links

B.2 PCTE object type hierarchy

-- CDIF: AttributableMetaObject Hierarchy: meta-entities
import object type system-object as RootEntity; -- root of hierarchy for CDIF object types
    AbstractionLevel : child type of RootEntity;
    AlternateName : child type of RootEntity;
    PresentationInformationObject : child type of RootEntity;
    SemanticInformationObject : child type of RootEntity;
        DataObject : child type of SemanticInformationObject;
        Derivation : child type of SemanticInformationObject;
        ProcessObject : child type of SemanticInformationObject;
    TextualConstraint : child type of RootEntity;
    ToolUser : child type of RootEntity;

B.3 PCTE common attribute types

Name : string;
BriefDescription : string;
FullDescription : string; -- Text

B.4 PCTE link types

-- CDIF: RootEntity.CreatedBy.ToolUser
CreatedBy : reference link to ToolUser;

```
-- CDIF: RootEntity.Has.AlternateName
 Has       : reference link to AlternateName;

-- CDIF: RootEntity_LastUpdatedBy_ToolUser
LastUpdatedBy    : reference link to ToolUser;

-- CDIF: RootEntity_Uses_AlternateName
Uses      : reference link to AlternateName;

-- CDIF: SemanticInformationObject.IsCategorizedIn.AbstractionLevel
IsCategorizedIn    : reference link to AbstractionLevel;

-- CDIF: SemanticInformationObject.ProducedBy.Derivation
ProducedBy  : reference link to Derivation;

-- CDIF: SemanticInformationObject.UsedIn.Derivation
UsedIn   : reference link to Derivation;

-- CDIF: TextualConstraint.IsConstraintOn.SemanticInformationObject
IsConstraintOn     : reference link to SemanticInformationObject;
```

B.5 PCTE object type extensions

```
-- CDIF: AbstractionLevel
extend object type AbstractionLevel with
attribute
    Name;                  -- Enumerated (Conceptual, Logical, Physical)
end AbstractionLevel;

-- CDIF: AlternateName
extend object type AlternateName with
attribute
    OtherLongName          : string;
    OtherName              : string;
end AlternateName;

-- CDIF: DataObject
extend object type DataObject with
```

```
attribute
    Name;
end DataObject;

-- CDIF: Derivation
extend object type Derivation with
attribute
    DerivationLanguage      : enumeration (Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other);
    DerivationText          : string;    -- Text
    IsRealizationOf         : boolean;
end Derivation;

-- CDIF: PresentationInformationObject

-- CDIF: ProcessObject
extend object type ProcessObject with
attribute
    ExecutionTimeInterval       : float;
    ExecutionTimeUnit           : enumeration (Picosecond, Nanosecond, Microsecond, Millisecond, Second,
                                    Minute, Hour, Day, Week, Month, Year);
    Name;
    SpecificationLanguage       : enumeration (Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL,
                                    Other);
    SpecificationText           : string; -- Text
end ProcessObject;

-- CDIF: RootEntity
extend object type RootEntity with
link
    CreatedBy;
    Has;
    LastUpdatedBy;
    Uses;
end RootEntity;

--- CDIF: SemanticInformationObject
extend object type SemanticInformationObject with
attribute
    BriefDescription;
```

```
    FullDescription;
link
    IsCategorizedIn;
    ProducedBy;
    UsedIn;
end SemanticInformationObject;

-- CDIF: TextualConstraint
extend object type TextualConstraint with
attribute
    BriefDescription;
    ConstraintExpression        : string;    -- Text
    ConstraintLanguage          : enumeration (Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other);
    FullDescription;
link
    IsConstraintOn;
end TextualConstraint;

-- CDIF: ToolUser
extend object type ToolUser with
attribute
    FullName                : string;
    SystemName              : string;
end ToolUser;

end CDIF_Common;
```

Annex C

(informative)

SDS CDIF_DataDefinition

C.1 Introductory elements

sds CDIF_ DataDefinition:

-- This annex defines the PCTE SDS CDIF_ DataDefinition that corresponds to
-- the CDIF subject area DataDefinition version 01.00 defined in
-- EIA/IS-113, CDIF - Integrated Meta-model / Data Definition Subject Area, May 1996 (draft)

import attribute type system-number; -- required for link keys
import attribute type system-system_key; -- may be required for creating implicit reverse links

C.2 PCTE object type hierarchy

-- CDIF: AttributableMetaObject Hierarchy: meta-entities
import object type system-object as RootEntity; -- root of hierarchy for CDIF object types
SemanticInformationObject; : child type of RootEntity
ComponentObject; : child type of SemanticInformationObject
Attribute; : child type of ComponentObject
ProjectedAttribute; : child type of Attribute
EquivalenceSet : child type of ComponentObject;
ReferencedElement : child type of ComponentObject;
DefinitionObject; : child type of SemanticInformationObject
DataType : child type of DefinitionObject;
AggregateDataType : child type of DataType;
BasicDataType : child type of DataType;
BinaryType : child type of BasicDataType;
FixedLengthBinaryType : child type of BinaryType;
VariableLengthBinaryType : child type of BinaryType;
BooleanType : child type of BasicDataType;
EnumerationType : child type of BasicDataType;
MagnitudeType : child type of BasicDataType;
MoneyType : child type of MagnitudeType;
NumericType : child type of MagnitudeType;

ApproximateNumericType : child type of NumericType;
ComplexType : child type of NumericType;
CartesianComplexType : child type of ComplexType;
PolarComplexType : child type of ComplexType;
ExactNumericType : child type of NumericType;
IntegerType : child type of ExactNumericType;
FixedDecimalType : child type of IntegerType;
BinaryCodedDecimalType : child type of FixedDecimalType;
PackedDecimalType : child type of FixedDecimalType;
SerialType : child type of IntegerType;
TemporalType : child type of MagnitudeType;
DateType : child type of TemporalType;
TimeIntervalType : child type of TemporalType;
DayTimeIntervalType : child type of TimeIntervalType;
YearMonthIntervalType : child type of TimeIntervalType;
TimeStampType : child type of TemporalType;
TimeType : child type of TemporalType;
StringType : child type of BasicDataType;
FixedLengthStringType : child type of StringType;
NLFixedLengthStringType : child type of FixedLengthStringType;
VariableLengthStringType : child type of StringType;
NLVariableLengthStringType : child type of VariableLengthStringType;
VoidType : child type of BasicDataType;
QualifiedDataType : child type of DataType;
RefinedDataType : child type of DataType;
Qualifier : child type of SemanticInformationObject;
ArrayQualifier : child type of Qualifier;
BoundedArrayQualifier : child type of ArrayQualifier;
UnboundedArrayQualifier : child type of ArrayQualifier;
PointerQualifier : child type of Qualifier;
Unit : child type of SemanticInformationObject;
ValueDomain : child type of SemanticInformationObject;
ValueDomainEnumeration : child type of ValueDomain;
ValueDomainProcedure : child type of ValueDomain;
ValueDomainRange : child type of ValueDomain;
ValueDomainRule : child type of ValueDomain;
ValueDomainGroup : child type of ValueDomain;

C.3 PCTE common attribute types

```
BitsPerCharacter        : integer;
IsLocal                 : boolean;
Length                  : natural;   -- Integer
LengthMultiplier        : enumeration (Bit, Byte, Kilobyte, Megabyte, Gigabyte);
MaxLength               : natural;   -- Integer
Name                    : string;
Operator                : enumeration (AND, OR, XOR, NOT);
Precision               : integer;
Scale                   : integer;
SpecificationLanguage   : enumeration (Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other);
SpecificationText       : string; -- Text
TimeZoneHours           : integer;
TimeZoneMinutes         : integer;
```

C.4 PCTE link types

```
-- CDIF: ArrayQualifier.HasType.DataType
HasType      : reference link [0..1] to DataType;

-- CDIF: ComponentObject.References.DefinitionObject
References   : reference link [0..1] to DefinitionObject;

-- CDIF: DataType.TakesValueFrom.ValueDomainGroup
TakesValueFrom      : reference link [0..1] to ValueDomainGroup;

-- CDIF: DefinitionObject.Contains.ComponentObject
import link type CDIF_DataModeling-Contains;
extend link type Contains to ComponentObject;

-- CDIF: EquivalenceSet.HasMember.ComponentObject
HasMember           : reference link [2..] (number) to ComponentObject;

-- CDIF: NumericType.IsMeasuredIn.Unit
IsMeasuredIn        : reference link [0..1] to Unit;

-- CDIF: ProjectedAttribute.IsProjectionOf.Attribute
IsProjectionOf      : reference link [0..] (number) to Attribute;
```

```
-- CDIF: QualifiedDataType.IsQualificationOf.DataType
IsQualificationOf      : reference link [0..1] to DataType;

-- CDIF: QualifiedDataType.IsQualifiedBy.Qualifier
IsQualifiedBy          : reference link [0..] (number) to Qualifier;

-- CDIF: ReferencedElement.DefinesPath.ComponentObject
DefinesPath            : reference link [1..] (number) to ComponentObject;

-- CDIF: RefinedDataType.IsRefinementOf.DataType
IsRefinementOf         : reference link [0..1] to DataType;

-- CDIF: ValueDomainGroup.Contains.ValueDomain
extend link type Contains to ValueDomain;
```

C.5 PCTE object type extensions

```
-- CDIF: AggregateDataType
-- No extension

-- CDIF: ApproximateNumericType
extend object type ApproximateNumericType with
attribute
    Precision;
    Scale;
end ApproximateNumericType;

-- CDIF: ArrayQualifier
extend object type ArrayQualifier with
link
    HasType;
end ArrayQualifier;

-- CDIF: Attribute
extend object type Attribute with
attribute
    DefaultValue       : string;
    IsOptional         : boolean;
```

```
    Name;
end Attribute;

-- CDIF: BasicDataType
-- No extension

-- CDIF: BinaryCodedDecimalType
-- No extension

-- CDIF: BinaryType
-- No extension

-- CDIF: BooleanType
-- No extension

-- CDIF: BoundedArrayQualifier
extend object type BoundedArrayQualifier with
attribute
    MaxSubscript     : integer;
    MinSubscript     : integer;
end BoundedArrayQualifier;

-- CDIF: CartesianComplexType
extend object type CartesianComplexType with
attribute
    Precision;
    Scale;
end CartesianComplexType;

-- CDIF: ComplexType
-- No extension

-- CDIF: ComponentObject
extend object type ComponentObject with
link
    References;
end ComponentObject;
```

```
-- CDIF: DataType
extend object type DataType with
attribute
    FormatStringLanguage     : enumeration (Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other);
    FormatStringValue        : string;
    Usage                    : string;      -- Text
link
    TakesValueFrom;
end DataType;

-- CDIF: DateType
-- No extension

-- CDIF: DayTimeIntervalType
-- No extension

-- CDIF: DefinitionObject
extend object type DefinitionObject with
attribute
    Name;
    Operator;              -- Enumerated (AND, XOR)
    SpecificationLanguage;
    SpecificationText;    -- Text
link
    Contains;
    IsConstructedWith;
end DefinitionObject;

-- CDIF: EnumerationType
-- No extension

-- CDIF: EquivalenceSet
extend object type EquivalenceSet with
link
    HasMember;
end EquivalenceSet;
```

```
-- CDIF: ExactNumericType
-- No extension

-- CDIF: FixedDecimalType
extend object type FixedDecimalType with
attribute
    Precision;
    Scale;
end FixedDecimalType;

-- CDIF: FixedLengthBinaryType
extend object type FixedLengthBinaryType with
attribute
    Length;
    LengthMultiplier;
end FixedLengthBinaryType;

-- CDIF: FixedLengthStringType
extend object type FixedLengthStringType with
attribute
    Length;
    LengthMultiplier;
end FixedLengthStringType;

-- CDIF: IntegerType
extend object type IntegerType with
attribute
    SignedFlag    : boolean;
end IntegerType;

-- CDIF: MagnitudeType
-- No extension

-- CDIF: MoneyType
extend object type MoneyType with
attribute
    Currency      : string;
    Precision;
```

```
    Scale;
end MoneyType;

-- CDIF: NLFixedLengthStringType
extend object type NLFixedLengthStringType with
attribute
    BitsPerCharacter;
end NLFixedLengthStringType;

-- CDIF: NLVariableLengthStringType
extend object type NLVariableLengthStringType with
attribute
    BitsPerCharacter;
end NLVariableLengthStringType;

-- CDIF: NumericType
extend object type NumericType with
link
    IsMeasuredIn;
end NumericType;

-- CDIF: PackedDecimalType
-- No extension

-- CDIF: PointerQualifier
-- No extension

-- CDIF: PolarComplexType
extend object type PolarComplexType with
attribute
    DistancePrecision    : integer;
    DistanceScale        : integer;
    GradientPrecision    : integer;
    GradientScale        : integer;
end PolarComplexType;

-- CDIF: ProjectedAttribute
extend object type ProjectedAttribute with
```

```
attribute
    SpecificationLanguage;
    SpecificationText;    -- Text
link
    IsProjectionOf;
end ProjectedAttribute;

-- CDIF: QualifiedDataType
extend object type QualifiedDataType with
link
    IsQualificationOf;
    IsQualifiedBy;
end QualifiedDataType;

-- CDIF: Qualifier
extend object type Qualifier with
attribute
    PrecedenceNumber        : integer;
end Qualifier;

-- CDIF: ReferencedElement
extend object type ReferencedElement with
link
    DefinesPath;
end ReferencedElement;

-- CDIF: RefinedDataType
extend object type RefinedDataType with
link
    IsRefinementOf;
end RefinedDataType;

-- CDIF: SemanticInformationObject
extend object type SemanticInformationObject with
attribute
    BriefDescription;
    FullDescription;
end SemanticInformationObject;
```

```
-- CDIF: SerialType
extend object type SerialType with
attribute
    Cycle           : boolean;
    Interval        : integer;
    StartingValue   : integer;
end SerialType;

-- CDIF: StringType
extend object type StringType with
attribute
    CharacterSet    : integer;
    StringEncoding  : enumeration ("ISO-2022", "ISO-4873", "ISO-8825", "CCITT-T61")
end StringType;

-- CDIF: TemporalType
-- No extension

-- CDIF: TimeIntervalType
-- No extension

-- CDIF: TimeStampType
extend object type TimeStampType with
attribute
    IsLocal;
    TimeZoneHours;
    TimeZoneMinutes;
end TimeStampType;

-- CDIF: TimeType
extend object type TimeType with
attribute
    IsLocal;
    TimeZoneHours;
    TimeZoneMinutes;
end TimeType;
```

```
-- CDIF: UnboundedArrayQualifier
-- No extension

-- CDIF: Unit
extend object type Unit with
attribute
    ExponentForAmpere     : integer;
    ExponentForCandela    : integer;
    ExponentForKelvin     : integer;
    ExponentForKilogram   : integer;
    ExponentForMeter      : integer;
    ExponentForMole       : integer;
    ExponentForSecond     : integer;
    IsSI                  : boolean;
    Name;
end Unit;

-- CDIF: ValueDomain
extend object type ValueDomain with
attribute
    Name;
end ValueDomain;

-- CDIF: ValueDomainEnumeration
extend object type ValueDomainEnumeration with
attribute
    Value  : string;
end ValueDomainEnumeration;

-- CDIF: ValueDomainGroup
extend object type ValueDomainGroup with
attribute
    Name;
    Operator;
link
    Contains;
end ValueDomainGroup;
```

```
-- CDIF: ValueDomainProcedure
extend object type ValueDomainProcedure with
attribute
    ProcedureName       : string;
    SpecificationLanguage;
    SpecificationText;
end ValueDomainProcedure;

-- CDIF: ValueDomainRange
extend object type ValueDomainRange with
attribute
    HighValue           : string;
    HighValueIncluded   : boolean;
    LowValue            : string;
    LowValueIncluded    : boolean;
end ValueDomainRange;

-- CDIF: ValueDomainRule
extend object type ValueDomainRule with
attribute
    SpecificationLanguage;
    SpecificationString : string;
end ValueDomainRule;

-- CDIF: VariableLengthBinaryType
extend object type VariableLengthBinaryType with
attribute
    LengthMultiplier;
    MaxLength;
end VariableLengthBinaryType;

-- CDIF: VariableLengthStringType
extend object type VariableLengthStringType with
attribute
    LengthMultiplier;
    MaxLength;
end VariableLengthStringType;
```

```
-- CDIF: VoidType
-- No extension

-- CDIF: YearMonthIntervalType
-- No extension

end CDIF_DataDefinition;
```

Annex D

(informative)

SDS CDIF_DataModelling

D.1 Introductory elements

sds CDIF_ DataModeling:

-- This annex defines the PCTE SDS CDIF_ DataModeling that corresponds to
-- the CDIF subject area DataModeling version 01.00 defined in
-- EIA/IS-114, CDIF - Integrated Meta-model / Data Modeling Subject Area, December 1996

import attribute type system-number;	-- required for link keys
import attribute type system-system_key;	-- may be required for creating implicit reverse links

D.2 PCTE object type hierarchy

-- CDIF: AttributableMetaObject Hierarchy: meta-entities	
import object type system-object as RootEntity;	-- root of hierarchy for CDIF object types
import object type CDIF_Common-SemanticInformationObject;	-- child type of RootEntity
AccessPath	: child type of SemanticInformationObject;
ComponentObject	: child type of SemanticInformationObject;
Attribute	: child type of ComponentObject;
ProjectedAttribute	: child type of Attribute;
DataModel	: child type of SemanticInformationObject;
DataModelObject	: child type of SemanticInformationObject;
-- Cluster	: child type of DataModelObject;
InheritableDataModelObject	: child type of DataModelObject;
-- Entity	: child type of InheritableDataModelObject;
-- Relationship	: child type of InheritableDataModelObject;
DataModelSubset	: child type of SemanticInformationObject;
DefinitionObject	: child type of SemanticInformationObject;
Cluster	: child type of DefinitionObject, DataModelObject;
Entity	: child type of DefinitionObject, InheritableDataModelObject;

Relationship : child type of DefinitionObject,
InheritableDataModelObject;
Role : child type of DefinitionObject;
RolePlayer : child type of DefinitionObject;
Key : child type of SemanticInformationObject;
CandidateKey : child type of Key;
ForeignKey : child type of Key;
ProjectionComponent : child type of SemanticInformationObject;
RoleConstraint : child type of SemanticInformationObject;
SubtypeSet : child type of SemanticInformationObject;
SubtypeSetMembershipCriterion : child type of SemanticInformationObject;

D.3 PCTE common attribute types

AvgNumberOfOccurrences : float;
DeletionTimePeriod : enumeration (Millisecond, Second, Minute, Hour, Day, Week, Month, Year);
InsertionTimePeriod : enumeration (Millisecond, Second, Minute, Hour, Day, Week, Month, Year);
MaxNumberOfOccurrences : natural; -- integer
MinNumberOfOccurrences : natural; -- integer
Name : string;
NumberOfDeletions : float;
NumberOfInsertions : float;
NumberOfReads : float;
NumberOfUpdates : float;
Operator : enumeration (AND, OR, XOR, NOT);
ReadTimePeriod : enumeration (Millisecond, Second, Minute, Hour, Day, Week, Month, Year);
SpecificationLanguage : enumeration (Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other);
SpecificationText : string; -- Text
UpdateTimePeriod : enumeration (Millisecond, Second, Minute, Hour, Day, Week, Month, Year);

D.4 PCTE link types

-- CDIF: AccessPath.Incorporates.Attribute
Incorporates : reference link [0..] (number) with
attribute
IsAscending : boolean;
SequenceNumber : natural; -- Integer
end Incorporates;
extend link type Incorporates to Attribute;

```
-- CDIF: AccessPath.Instantiates.Key
Instantiates          : reference link [0..1] to Key;

-- CDIF: Attribute.IsDiscriminatorFor.SubtypeSetMembershipCriterion
IsDiscriminatorFor    : reference link [0..] (number) to SubtypeSetMembershipCriterion;

-- CDIF: Attribute.IsInheritedFrom.Attribute
IsInheritedFrom       : reference link to Attribute;

-- CDIF: CandidateKey.Incorporates.ForeignKey
extend link type Incorporates to ForeignKey;

-- CDIF: Cluster.Collects.DataModelObject
Collects                    : reference link [0..] (number);
extend link type Collects to DataModelObject;

-- CDIF: DataModel.Collects.DataModelObject
-- extend link type Collects to DataModelObject;

-- CDIF: DataModelObject.ActsAs.RolePlayer
ActsAs                      : reference link [0..] (number) to RolePlayer;

-- CDIF: DataModelObject.IsMemberOf.DataModelSubset
IsMemberOf                  : reference link [0..] (number) to DataModelSubset;

-- CDIF: DataModelSubset.Excludes.Attribute
Excludes                    : reference link [0..] (number) to Attribute;

-- CDIF: DataModelSubset.IsSubsetOf.DataModel
IsSubsetOf                  : reference link [1..1] to DataModel;

-- CDIF: DefinitionObject.Contains.ComponentObject
Contains                    : reference link [0..] (number) to ComponentObject with
attribute
    SequenceNumber;
end Contains;
```

```
-- CDIF: DefinitionObject.IsConstructedWith.ProjectionComponent
IsConstructedWith          : reference link [0..] (number) to ProjectionComponent with
attribute
    SequenceNumber;
end IsConstructedWith;

-- CDIF: Entity.IsIdentifiedBy.Key
IsIdentifiedBy             : reference link [0..] (number) to Key;

-- CDIF: Entity.IsAccessedUsing.AccessPath
IsAccessedUsing            : reference link [0..] (number) to AccessPath;

-- CDIF: ForeignKey.Incorporates.RolePlayer
extend link type Incorporates to RolePlayer;

-- CDIF: ForeignKey.References.CandidateKey
References                 : reference link [1..1] to CandidateKey;

-- CDIF: InheritableDataObject.IsSubtypeIn.SubtypeSet
IsSubtypeIn                : reference link [0..] (number) to SubtypeSet with
attribute
    SpecificationLanguage;
    SpecificationText;
    StoreWithSupertype     : boolean;
end IsSubtypeIn;

-- CDIF: InheritableDataObject.IsSupertypeFor.SubtypeSet
IsSupertypeFor             : reference link [0..] (number) to SubtypeSet;

-- CDIF: Key.Incorporates.Attribute
-- extend link type Incorporates to Attribute;

-- CDIF: Key.Incorporates.SemanticInformationObject
extend link type Incorporates to SemanticInformationObject;

-- CDIF: ProjectedAttribute.IsProjectionOf.Attribute
IsProjectionOf             : reference link [0..] (number) with
```

```
attribute
    SequenceNumber;
end IsProjectionOf;
extend link type IsProjectionOf to Attribute;

-- CDIF: ProjectionComponent.IsFullProjectionOf.DefinitionObject
IsFullProjectionOf          : reference link [1..1] to DefinitionObject;

-- CDIF: ProjectionComponent.IsProjectionOf.Attribute
-- extend link type IsProjectionOf to Attribute;

-- CDIF: Role.BelongsTo.Relationship
BelongsTo                   : reference link [1..1] to Relationship;

-- CDIF: RoleConstraint.Incorporates.RoleConstraint
extend link type Incorporates to RoleConstraint;

-- CDIF: RoleConstraint.Incorporates.RolePlayer
-- extend link type Incorporates to RolePlayer;

-- CDIF: RoleConstraint.Incorporates.SemanticInformationObject
-- extend link type Incorporates to SemanticInformationObject;

-- CDIF: RolePlayer.IsSupportedBy.Key
IsSupportedBy               : reference link [0..1] to Key;

-- CDIF: RolePlayer.Plays.Role
Plays                       : reference link [0..1] to Role;

-- CDIF: RolePlayer.Refines.RolePlayer
Refines                     : reference link [0..1] to RolePlayer;

-- CDIF: RolePlayer.RefinesForSubtype.DataModelObject
RefinesForSubtype           : reference link [0..] (number) to DataModelObject;

-- CDIF: SubtypeSet.Specifies.SubtypeSetMembershipCriterion
Specifies                   : reference link [0..] (number) to SubtypeSetMembershipCriterion;
```

```
-- CDIF: SubtypeSetMembershipCriterion.Selects.InheritableDataModelObject
Selects                         : reference link [1..1] to InheritableDataModelObject;
```

D.5 PCTE object type extensions

```
-- CDIF: AccessPath
extend object type AccessPath with
attribute
    Name;
    SpecificationLanguage;
    SpecificationText;    -- Text
link
    Incorporates;
    Instantiates;
end AccessPath;

-- CDIF: Attribute
extend object type Attribute with
attribute
    DefaultValue      : string;
    IsOptional        : boolean;
    Name;
link
    IsDiscriminatorFor;
    IsInheritedFrom;
end Attribute;

-- CDIF: CandidateKey
extend object type CandidateKey with
attribute
    IsPrimary         : boolean;
link
    Incorporates;
end CandidateKey;

-- CDIF: Cluster
extend object type Cluster with
attribute
    Name;
```

```
-- not defined in C 5.3.4
link
    Collects;
end Cluster;

-- CDIF: ComponentObject
extend object type ComponentObject with
-- ComponentObject.References.DefinitionObject in meta-object hierarchy, but
-- defined in Data Definition.
end ComponentObject;

-- CDIF: DataModel
extend object type DataModel with
attribute
    ModelType    : string;
    Name;
link
    Collects;
end DataModel;

-- CDIF: DataModelObject
extend object type DataModelObject with
link
    ActsAs;
    IsMemberOf;
end DataModelObject;

-- CDIF: DataModelSubset
extend object type DataModelSubset with
attribute
    Name;
link
    Excludes;
    IsSubsetOf;
end DataModelSubset;
```

```
-- CDIF: DefinitionObject
extend object type DefinitionObject with
attribute
    Name;
    Operator;           -- Enumerated (AND, XOR)
    SpecificationLanguage;
    SpecificationText;    -- Text
link
    Contains;
    IsConstructedWith;
end DefinitionObject;

-- CDIF: Entity
extend object type Entity with
attribute
    AvgNumberOfOccurrences;
    DeletionTimePeriod;
    EntityType              : enumeration (Kernel, Characteristic, Associative);
    InsertionTimePeriod;
    MaxNumberOfOccurrences;
    MinNumberOfOccurrences;
    NormalizationState      : enumeration (UNF, "1NF", "2NF", "3NF", BCNF, "4NF", "5NF")
    NumberOfDeletions;
    NumberOfInsertions;
    NumberOfReads;
    NumberOfUpdates;
    ReadTimePeriod;
    UpdateTimePeriod;
    Usage                   : string;    -- Text
link
    IsIdentifiedBy;
    IsAccessedUsing;
end Entity;

-- CDIF: ForeignKey
extend object type ForeignKey with
link
    Incorporates;
```

```
    References;
end ForeignKey;

-- CDIF: InheritableDataModelObject
extend object type InheritableDataModelObject with
attribute
    IsAbstract      : boolean;
link
    IsSubtypeIn;
    IsSupertypeFor;
end InheritableDataModelObject;

-- CDIF: Key
extend object type Key with
attribute
    Name;
    SpecificationLanguage;
    SpecificationText;    -- Text
link
    Incorporates;
end Key;

-- CDIF: ProjectedAttribute
extend object type ProjectedAttribute with
attribute
    SpecificationLanguage;
    SpecificationText;    -- Text
link
    IsProjectionOf;
end ProjectedAttribute;

-- CDIF: ProjectionComponent
extend object type ProjectionComponent with
attribute
    Name;
    SpecificationLanguage;
    SpecificationText; -- Text
```

```
link
    IsFullProjectionOf;
    IsProjectionOf;
end ProjectionComponent;

-- CDIF: Relationship
extend object type Relationship with
attribute
    InverseName      : string;
end Relationship;

-- CDIF: Role
extend object type Role with
attribute
    IsMaster   : boolean;
    IsSource   : boolean;
link
    BelongsTo;
end Role;

-- CDIF: RoleConstraint
extend object type RoleConstraint with
attribute
    Name;
    Operator;  -- Enumerated (AND, OR, XOR)
link
    Incorporates;
end RoleConstraint;

-- CDIF: RolePlayer
extend object type RolePlayer with
attribute
    AvgNumberOfOccurrences;
    DeleteEffect             : enumeration (RESTRICTS, CASCADES, SETNULL, SETDEFAULT);
    DeletionTimePeriod;
    InsertEffect             : enumeration (RESTRICTS, CASCADES, SETNULL, SETDEFAULT);
    InsertionTimePeriod;
    IsDeleteDeferrable       : boolean;
```

```
    IsInsertDeferrable          : boolean;
    IsUpdateDeferrable          : boolean;
    MaxInnerCardinality         : string;
    MaxNumberOfOccurrences;
    MaxOuterCardinality         : string;
    MinInnerCardinality         : string;
    MinNumberOfOccurrences;
    MinOuterCardinality         : string;
    NumberOfDeletions;
    NumberOfInsertions;
    NumberOfReads;
    NumberOfUpdates;
    ReadTimePeriod;
    UpdateEffect                : enumeration (RESTRICTS, CASCADES, SETNULL, SETDEFAULT);
    UpdateTimePeriod;
link
    IsSupportedBy;
    Plays;
    Refines;
    RefinesForSubtype;
end RolePlayer;

-- CDIF: SemanticInformationObject
extend object type SemanticInformationObject with
attribute
    BriefDescription;
    FullDescription;
end SemanticInformationObject;

-- CDIF: SubtypeSet
extend object type SubtypeSet with
attribute
    IsExclusive    : boolean;
    Name;
    SubtypeListIsClosed      : boolean;
link
    Specifies;
end SubtypeSet;
```

```
-- CDIF: SubtypeSetMembershipCriterion
extend object type SubtypeSetMembershipCriterion with
attribute
    DiscriminatorValue      : string;
    SpecificationLanguage;
    SpecificationText;      -- Text
link
    Selects;
end SubtypeSetMembershipCriterion;

end CDIF_DataModeling;
```

Printed copies can be ordered from:

ECMA
114 Rue du Rhône
CH-1204 Geneva
Switzerland

Fax: +41 22 849.60.01
Internet: documents@ecma.ch

Files can be downloaded from our FTP site, **ftp.ecma.ch,** logging in as **anonymous** and giving your E-mail address as **password**. This Standard is available from library **ECMA-ST** as a compacted, self-expanding file in MSWord 6.0 format (file E270-DOC.EXE) and as an Acrobat PDF file (file E270-PDF.PDF). File E270-EXP.TXT gives a short presentation of the Standard.

Our web site, http://www.ecma.ch, gives full information on ECMA, ECMA activities, ECMA Standards and Technical Reports.

ECMA

114 Rue du Rhône
CH-1204 Geneva
Switzerland