INTERNATIONAL STANDARD

ISO/IEC 14478-2

First edition
1998-12-15

Corrected and reprinted
2000-09-15

Information technology — Computer graphics and image processing — Presentation Environment for Multimedia Objects (PREMO) —

Part 2: Foundation Component

Technologies de l'information — Infographie et traitement d'images — Environnement de présentation d'objects multimédia (PREMO) —

Partie 2: Composant fondamental

Reference number
ISO/IEC 14478-2:1998(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 14478 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14478-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 24, *Computer graphics and image processing*.

ISO/IEC 14478 consists of the following parts, under the general title *Information technology — Computer graphics and image processing — Presentation Environment for Multimedia Objects (PREMO)*:

— *Part 1: Fundamentals of PREMO*

— *Part 2: Foundation Component*

— *Part 3: Multimedia Systems Services*

— *Part 4: Modelling, rendering and interaction component*

Annexes A and B form a normative part of this part of ISO/IEC 14478. Annex C is for information only.

Introduction

This part of ISO/IEC 14478 defines those object types and non–object types which belong to the Foundation Component. Any conforming PREMO implementation shall support these object types. The description of object types categories are given first and then the foundation object types in each category are described.

Information technology — Computer graphics and image processing — Presentation Environment for Multimedia Objects (PREMO) — Part 2: Foundation Component

1 Scope

This part of ISO/IEC 14478 lists an initial set of object types and non–object types useful for the construction of, presentation of, and interaction with multimedia information. This part is dependent on the PREMO object model defined in clause 8 of ISO/IEC 14478-1. The foundation component does not depend on any other components.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 14478. 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 14478 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 14478-1:1998, *Information technology — Computer graphics and image processing — Presentation Environment for Multimedia Objects (PREMO) — Part 1: Fundamentals of PREMO.*

ISO/IEC 11172 (all parts), *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s.*

3 Definitions

3.1 PREMO Part 1 definitions

This part of ISO/IEC 14478 makes use of all terms defined in ISO/IEC 14478-1 (Fundamentals of PREMO).

3.2 Additional definitions

For the purposes of this part of ISO/IEC 14478, the following definitions apply.

3.2.1 basic data type: Non-object data type which cannot be expressed via other data types. Examples are integers, floating point numbers.

3.2.2 constructed data type: As opposed to basic data type; non-object data type which is constructed with the help of permitted type constructors using basic data types.

3.2.3 time: A non–object data type which is appropriate for the representation of real time in the execution environment. It is typically realized through either float numbers or large (64 bit) integers.

3.2.4 extended coordinates: An extension of real, integer, or time coordinates with the symbols $-\infty$ and ∞, and the natural comparison operators. It gives a succinct way of describing unlimited intervals on these coordinate systems.

3.2.5 key–value pair: A constructed data type, consisting of a key (described as a string) and a corresponding value.

3.2.6 foundation object type: Object types defined in the foundation component of PREMO.

3.2.7 structure: A category of object types in PREMO; these objects are characterized through attributes only.

3.2.7.1 structure tag: A synonym for an attribute for a structure.

3.2.8 property: Key with associated value or sequence of values, which can be attached to any PREMO object, and which can be inquired, possibly created and deleted through operations defined on the object.

3.2.8.1 read–only property: A property whose value or values cannot be set by operations of the object.

3.2.9 fundamental object behaviour: Operations defined on the *PREMOObject* type; this type is the supertype of all PREMO object types.

3.2.10 finite state machine: Implementation of an abstract finite state automaton.

3.2.11 constraint: A constructed data type, consisting of a key-value pair and an associated constraint operation; this latter is used to compare the values, in case the keys are identical.

3.2.12 event: A constructed data type, serving as a basic building block for the PREMO Event Model.

3.2.12.1 event source: Object (instance) which creates events. This is a structure tag of an event.

3.2.12.2 event client: Object (instance) which consumes events.

3.2.12.3 event name: A means to denote and/or to refer to a specific event. This name is also referred to as *event type*. This is a structure tag of an event.

3.2.12.4 event data: List of non-object types in the form of key–value pairs attached to an event. This is a structure tag of an event.

3.2.13 event handler: An object which provides event processing services to other objects.

3.2.14 era: The base date for all PREMO systems to measure the amount of elapsed time. This value is set to 00:00am, 1st January 1970, UTC.

3.2.15 reference point: A point in the internal coordinate system of a synchronizable objects, to which a synchronization element is attached.

3.2.16 synchronization element: Synchronization information for a synchronizable object; it contains information on another object and its operation which shall be invoked if synchronization is set up.

3.3.17 capability: Description of the property values an object type can take for a specific key.

3.3.18 native property value: Description of the property value an object instance can take for a specific key.

3.3.19 private properties: Properties of the object which are not defined as part of the functional specification of the object.

The following alphabetical list gives the sub-clause of each definition.

basic data type	3.2.1
capability	3.3.17
constraint	3.2.11
constructed data type	3.2.2
event	3.2.12
event client	3.2.12.2
event data	3.2.12.4
event handler	3.2.13

event name	3.2.12.3
event source	3.2.12.1
event type	3.2.12.3
era	3.2.14
extended coordinates	3.2.4
finite state machine	3.2.10
foundation object type	3.2.6
fundamental object behaviour	3.2.9
key–value pair	3.2.5
native property value	3.3.18
private properties	3.3.19
property	3.2.8
read–only property	3.2.8.1
reference point	3.2.15
structure	3.2.7
structure tag	3.2.7.1
synchronization element	3.2.16
time	3.2.3

4 Symbols and abbreviations

AIFF:	Audio Interchange File Format.
FSM:	Finite State Machine.
IEC:	International Electrotechnical Commission.
IS:	International Standard.
ISO:	International Organization for Standardization.
MPEG:	Moving Picture Experts Group.
PREMO:	Presentation Environments for Multimedia Objects.
2D:	Two-dimensional.
3D:	Three-dimensional.

5 Conformance

A conforming implementation of the PREMO Foundation Component shall comply with the general conformance rules defined in clause 5 of ISO/IEC 14478-1 and the component specification in clause 10.

6 Foundation non-object types

The foundation non–object types in PREMO are defined in two categories: basic data types, and data types directly defined from these basic data types in terms of the notations described in clause A.2 of ISO/IEC 14478-1.

The basic data types are (with their type names):

a) **N**: non-negative integer.

b) **Z**: integer.

c) **R**: real number.

d) *ObjectType*: a data type uniquely identifying an object type.

e) *EventId*: a data type uniquely identifying an event registration for a PREMO event handler.

f) *Time*: a data type to measure progression of real world time. This type is either a real number or a (possibly large) integer. The choice among these is implementation dependent.

g) As described in 8.5 of ISO/IEC 14478-1, for each object of type *T* an object reference type, which is a non–object type, referring to object instances of type *T*, automatically exists in PREMO. As a notational convention, *RefT* denotes the non–object type of object reference referring to object instances of type *T*.

The environment shall provide comparison facilities for each basic data type which unambiguously decide whether two data values are identical or not. In the case of object references the environment shall also include a facility to test whether two references refer to the same object instance or not, or whether the value of the object reference is *NULLObject*. How these facilities are realized depends on the programming language and the execution environment in which PREMO is implemented.

Coordinate spaces can be “extended” to include positive and negative “infinity”. Although the underlying implementation may not have a direct representation of these types, the obvious extension of the notion of “greater than”, “smaller than”, etc., on these types allows the behaviour of objects to be defined more succinctly. The following extended coordinate space definitions are used:

h) Extended real numbers:

$$R_{\infty} == \mathbf{R} \cup \{-\infty, \infty\}$$

i) Extended integers:

$$Z_{\infty} == \mathbf{Z} \cup \{-\infty, \infty\}$$

j) Extended time:

$$Time_{\infty} == Time \cup \{-\infty, \infty\}$$

The foundation object types, described in this part, make also use of a number of (constructed) non–object types, defined formally in 9.2 (page 27). Some of these non–object types play a key role in the behavioural description of several object types; they are therefore also listed here, to make the semantic description in clause 7 easier to follow.

— Boolean:

$$Boolean ::= TRUE \mid FALSE$$

— Character String:

$$String ::= \mathrm{seq}\ Char$$

— Constraint specification for key–value pairs (used, for example, by property management, event handlers, and aggregate object types):

$$\begin{aligned} ConstraintOp ::= & \ Equal \mid NotEqual \\ & \mid GreaterThan \mid GreaterThanOrEqual \mid LessThan \mid LessThanOrEqual \\ & \mid Prefix \mid Suffix \mid NotPrefix \mid NotSuffix \\ & \mid Includes \mid Excludes \end{aligned}$$

Values in an operation request are constrained to values which satisfy these type constraints and the constructions defined in clause A.2 of ISO/IEC 14478-1 (see also 8.6 of ISO/IEC 14478-1). No particular representation for these values is mandated by the PREMO functional specification, although bindings of PREMO to programming languages or to distributed programming paradigms may specify such formats.

7 Foundation object types

7.1 Introduction

Foundation objects types are those which support a fundamental set of services suitable for use by a wide variety of higher level components. PREMO conformance rules require that, whenever a PREMO implementation includes these objects, they be included in the manner specified in this clause. This is the basis for interoperability. The following criteria are used to identify foundation objects:

a) they are used by a majority of higher level components;
b) together they provide an adequate minimal functional set;
c) they are needed to support output on widely available presentation resources;
d) algorithms exist for decomposing more complex functionality into the foundation object types.

In this clause, foundation object types are identified. By means of subtyping the application developer or component supplier may create objects and object types for their own specific needs. Clause 9 of this part gives the detailed definitions of each of these object types; clause A gives an pictorial overview of all object types defined in this clause.

7.2 PREMO objects and fundamental object behaviour

All PREMO objects are assumed to be subtyped from a type called *PREMOObject*. *PREMOObject* is an abstract type, i.e., it is not instantiable.

Operations on *PREMOObject* type fall into two categories described below.

7.2.1 Creation and destruction of objects

These operations are used by the object and object reference life cycle facilities when object instances are created and destroyed (see 8.11 of ISO/IEC 14478-1 for a detailed description of these facilities). The *initialize*, *initializeOnCopy*, and *destruct* operations are defined to be protected, i.e., no other PREMO object can re–initialize a PREMO object or directly call the *destruct* operation, only through the facilities provided by the environment.

7.2.2 Inquiries on types

These operations return information on the object type, the sequence of supertypes, or the complete type graph of the object. Using the information returned by these operations, complex negotiations are possible to optimize the behaviour of various other PREMO objects.

7.3 Simple PREMO objects

SimplePREMOObject is an abstract subtype of *PREMOObject*. *SimplePREMOObject* does not extend the behaviour of *PREMOObject*, but serves as a common supertype for a family of PREMO objects, called structures. Using such a supertype allows operation specifications to impose type constraints on their arguments.

7.3.1 Structures

The term “structure” does not denote a specific object type in PREMO but, instead, a category of types. These object types are characterized by:

a) they are the subtypes of *SimplePREMOObject* but are *not* subtypes of *EnhancedPREMOObject;*

b) they are not abstract types, although they may be generic types;

c) their behaviour in PREMO is expressed in terms of attributes rather than explicit operations (apart from the operations inherited from the supertype *PREMOObject*).

The attributes of a structure are also referred to as “structure tags”.

NOTE — Implementations, or further components, may define subtypes of structures by adding operations to the type specification. Item c above does *not* preclude this. However, as a use of terminology, such types are not labelled as “structures” any more.

Structures can be used as tools to encapsulate various non–object data into the object hierarchy. As an example, the following object type is used to describe constraints on key–value pairs:

```
╒══ Constraint ══════════════════════════════
│
   SimplePREMOObject
  │
  │ constraintOp: ConstraintOp
  │ kv: Key × Value
  │───────────────────────
  │
╘══ Constraint ══════════════════════════════
```

(This structure, formally defined in 9.5, plays an important role in the behavioural description of various objects in PREMO.)

NOTE — To increase the efficiency of the implementations, some programming languages may choose to implement structures as special data types and not as objects.

One of the most important structures used in PREMO is the event structure. Events structures consist of the following structure tags (see 9.5 for the precise specifications): an *event name* that provides a means to denote or refer to the event, also referred to as *event type*, an *event data* which is a sequence of key–value pairs, and the *event source*, which is the reference to the object instance which has created this event.

7.4 Callback objects

Very often object instances have to be notified by other objects on some status change, event occurrences, etc. This is done by ‘registering interest’ in some events. PREMO defines an abstract type, called *Callback*, to facilitate such mechanisms.

The *Callback* object type defines one single asynchronous operation, called *callback*. The signature of this operation consists of one input argument, which is a reference to an *Event* structure (see 9.5.2 for a detailed specification of this structure). Various PREMO objects, which may have to be notified under various circumstances, are defined to be subtypes of *Callback*, defining a type–specific behaviour to the *callback* operation.

NOTE — A typical example for the usage of the callback mechanism is the PREMO Event Model, described in detail in 7.7.1.

Whereas, in simple cases, the semantics of the *callback* operation may be defined to affect the state of the object directly, it is very often the case that this operation acts only as an entry point to call other operations on the object. To facilitate this second case, PREMO also defines a subtype of *Callback*, called *CallbackByName*. The (inherited) asynchronous *callback* operation of *CallbackByName* has the following behaviour: the *eventName* structure tag of the *Event* structure (appearing as the input argument of *callback*) is interpreted to be the name of a local operation which is then internally invoked by the *callback* operation. By default, all other structure tags of the *Event* structure are disregarded by the *callback* operation; subtypes of *CallbackByName* may add an additional behaviour to the operation which also takes these tags into consideration.

7.5 Enhanced PREMO Objects

EnhancedPREMOObject is an abstract type, i.e., is not instantiable. This type describes a set of behaviour, referred to as the *enhanced object behaviour*. The operations on *EnhancedPREMOObject* are related to *object properties*.

Enhanced PREMOObject represents the common, abstract supertype for PREMO objects with a more complex behaviour than, for example, structures. An important restriction in PREMO, which also reflects this characterization, is that only subtypes of *EnhancedPREMOObject* can appear in the ***provides service*** sub–schemas of profile specification (see clause 9 of ISO/IEC 14478-1).

7.5.1 Object properties

Properties are used to store values with an object that may be dynamically defined and are outside of the type system. Properties are pairs of keys and a sequence of values[1)] which are conceptually stored within a PREMO object. Operations are introduced to define, undefine, and inquire properties on PREMO object instances. Because, in general, the same key refers to a sequence of possible values, operations are also defined to add and to delete values from a sequence associated with a key. Properties can be used to implement various naming mechanisms, store information on the location of the object in a network, create annotations on object instances, etc.

Properties may be defined as read only. This means that they cannot be defined through an operation on the object, nor can they, or their associated values, be changed or deleted. Read only properties are typically set by the object when being initialized, and are used to describe the various capabilities of the object.

Properties of an object can also be matched against another list of key–value pairs using the *matchProperties* operation. This operation accepts a sequence of constraints, each defining a sequence of possible values for a specific property key, and returns the sequence of satisfied and unsatisfied constraints. Satisfaction is based on the boolean operation defined by the non–object data type *ConstraintOp*, see 9.2 (page 27), where the left operand of the operation is the value stored in the object, and the right operand of the operation is the value appearing in the argument of *matchProperties* (if the operation does not make sense, the result of the comparison is *FALSE*, i.e., it is the client's responsibility to ensure that the arguments are comparable). This mechanism may be used as part of complex negotiations.

NOTE — An example of using property matching is identifying the possible file formats of an audio service. The object providing the service may define a (read–only) sequence of values for the key "*AudioFormatK*", e.g., <"*AIFF*", "*IRCAM*">, describing the file formats it can use. The *matchProperties* operation may be invoked with a pair consisting of a key and a value, e.g.,

["*AudioFormatK*", "*AIFF*"]

using the comparison operator "Equal". The result will be:

satisfied: ["*AudioFormatK*", <"*AIFF*">]
unsatisfied: ["*AudioFormatK*", <"*IRCAM*">]

Another call, using:

["*AudioFormatK*", "*IRCAM*"]

1) A sequence may have only one element