INTERNATIONAL STANDARD

ISO/IEC 23005-7

Second edition
2014-01-15

Information technology — Media context and control —

Part 7:
Conformance and reference software

Technologies de l'information — Contrôle et contexte de supports —

Partie 7: Conformité et logiciel de référence

iTeh STANDARD PREVIEW
(standards.iteh.ai)

ISO/IEC 23005-7:2014
https://standards.iteh.ai/catalog/standards/sist/d2b31d54-1c32-4317-98f9-
f60407f84ea8/iso-iec-23005-7-2014

COPYRIGHT PROTECTED DOCUMENT

Contents

iTeh STANDARD PREVIEW
(standards.iteh.ai)

ISO/IEC 23005-7:2014
https://standards.iteh.ai/catalog/standards/sist/d2b31d54-1c32-4317-98f9-
f60407f84ea8/iso-iec-23005-7-2014

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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 document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 23005-7 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 23005-7:2011), which has been technically revised.

ISO/IEC 23005 consists of the following parts, under the general title *Information technology — Media context and control*:

— *Part 1: Architecture*

— *Part 2: Control information*

— *Part 3: Sensory information*

— *Part 4: Virtual world object characteristics*

— *Part 5: Data formats for interaction devices*

— *Part 6: Common types and tools*

— *Part 7: Conformance and reference software*

Introduction

This part of ISO/IEC 23005 specifies the conformance and reference software. The conformance and reference software of ISO/IEC 23005 serves three main purposes:

— validation of the written specification of the several parts of ISO/IEC 23005;

— clarification of the written specification of the several parts of ISO/IEC 23005; and

— conformance testing for checking interoperability for the various applications against the reference software which aims to be compliant with ISO/IEC 23005.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

iTeh STANDARD PREVIEW
(standards.iteh.ai)

Information technology — Media context and control —

Part 7:
Conformance and reference software

1 Scope

This part of ISO/IEC 23005 specifies the conformance and reference software implementing the normative clauses of all parts of ISO/IEC 23005. The information provided is applicable for determining the reference software modules available for the parts of ISO/IEC 23005, understanding the functionality of the available reference software modules, and utilizing the available reference software modules. The available reference software modules are specified in the form of application programming interfaces (API) according to ISO/IEC 23006-1.

Furthermore, this part of ISO/IEC 23005 provides means for conformance testing, i.e. bit-streams – XML descriptions – that conform or do not conform to the normative clauses of the other parts of ISO/IEC 23005 and informative descriptions thereof.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 19757-3, *Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron*

ISO/IEC 23005-2, *Information technology — Media context and control — Control information*

ISO/IEC 23005-3, *Information technology — Media context and control — Sensory information*

ISO/IEC 23005-4, *Information technology — Media context and control — Virtual world object characteristics*

ISO/IEC 23005-5, *Information technology — Media context and control — Data formats for interaction devices*

ISO/IEC 23005-6, *Information technology — Media context and control — Common types and tools*

ISO/IEC 23006-1, *Information technology — Multimedia service platform technologies — Architecture*

3 Terms, definitions, and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 23006-1 apply.

3.2 Abbreviated terms

For the purposes of this document, the abbreviated terms given in the following apply.

API   application programming interface

CIM   control information metadata

MXM   MPEG extensible middleware

SEM   sensory effect metadata

VWOC   virtual world object characteristics

IIM   interaction information metadata

4   Reference software for ISO/IEC 23005

4.1 Introduction

This Clause specifies the reference software for ISO/IEC 23005. The ISO/IEC 23005 reference software is written in Java and follows following package structure:

Name	Definition
org	Java package name for reference software provided by organizations such as ISO/IEC, W3C, or similar.
org.iso	Java package name for reference software provided by ISO/IEC.
org.iso.mpeg	Java package name for reference software provided by ISO/IEC JTC 1/SC 29/WG 11.
org.iso.mpeg.mpegv	Java package name for reference software provided in the course of the development of ISO/IEC 23005. NOTE 1 Subsequent packages for the individual ISO/IEC 23005 parts use the uncapitalized abbreviations as defined in 3.2, e.g., — org.iso.mpeg.mpegv.cidl, org.iso.mpeg.mpegv.dcdv, org.iso.mpeg.mpegv.sapv, org.iso.mpeg.mpegv.scdv and org.iso.mpeg.mpegv.sepv for ISO/IEC 23005-2, — org.iso.mpeg.mpegv.sedl and org.iso.mpeg.mpegv.sev for ISO/IEC 23005-3, — org.iso.mpeg.mpegv.vwoc for ISO/IEC 23005-4, or — org.iso.mpeg.mpegv.iidl, org.iso.mpeg.mpegv.dcv and org.iso.mpeg.mpegv.siv for ISO/IEC 23005-5. — org.iso.mpeg.mpegv.ct for ISO/IEC 23005-6. NOTE 2 Code included within these packages can be generated automatically by using tools such as JAXB.

Name	Definition
`org.iso.mpeg.mpegv.{part}` `.binary`	Java package name for binary representation of reference software provided in the course of the development of ISO/IEC 23005. the `{part}` placeholder is replaced by the component of each part, i.e., `cidl`, `dvdv`, `sapv`, `scdv`, `sepv`, `sedl`, `sev`, `vwoc`, `iidl`, `dcv`, `siv` and `ct`.
`org.iso.mpeg.mxm`	Java package name for reference software provided in the course of the development of ISO/IEC 23006-1. The actual API is defined within this package.

4.2 ISO/IEC 23005-2 APIs

4.2.1　Introduction

This clause specifies the API to the ISO/IEC 23005-2 reference software. The API is defined in Java and follows following package structure:

Name	Definition
`org.iso.mpeg.mxm.engine.cimengine`	Java package name for API to the ISO/IEC 23005-2 reference software.

4.2.2　CIM engine

The CIMEngine interface defines the methods for operating over data structures as defined within ISO/IEC 23005-2. Classes implementing the CIM engine interface act as factories creating instances of classes performing the following functionalities:

— Classes to create data structures, by means of the CIM creation engine

— Classes to access data contained in data structures, by means of CIM parser engine

4.2.3　CIM creation

Creating CIM structures involves the following interfaces:

— CIMCreator: an interface defining the methods to create CIM structures.

4.2.4　CIM access

Accessing CIM structures involves the following interfaces:

— CIMParser: an interface defining the methods to parse CIM structures.

4.3 ISO/IEC 23005-3 APIs

4.3.1 Introduction

This clause specifies the API to the ISO/IEC 23005-3 reference software. The API is defined in Java and follows following package structure:

Name	Definition
`org.iso.mpeg.mxm.engine.semengine`	Java package name for API to the ISO/IEC 23005-3 reference software.

4.3.2 SEM engine

The SEMEngine interface defines the methods for operating over data structures as defined within ISO/IEC 23005-3. Classes implementing the SEM engine interface act as factories creating instances of classes performing the following functionalities:

— Classes to create data structures, by means of the SEM creation engine.

— Classes to access data contained in data structures, by means of SEM parser engine.

4.3.3 SEM creation

Creating SEM structures involves the following interfaces:

— SEMCreator: an interface defining the methods to create SEM structures.

4.3.4 SEM access

Accessing SEM structures involves the following interfaces:

— SEMParser: an interface defining the methods to parse SEM structures.

4.4 ISO/IEC 23005-4 APIs

4.4.1 Introduction

This clause specifies the API to the ISO/IEC 23005-4 reference software. The API is defined in Java and follows following package structure:

Name	Definition
`org.iso.mpeg.mxm.engine.vwocengine`	Java package name for API to the ISO/IEC 23005-4 reference software.

4.4.2 VWOC engine

The VWOCEngine interface defines the methods for operating over data structures as defined within ISO/IEC 23005-4. Classes implementing the VWOC engine interface act as factories creating instances of classes performing the following functionalities:

⎯ Classes to create data structures, by means of the VWOC creation engine.

⎯ Classes to access data contained in data structures, by means of VWOC parser engine.

4.4.3 VWOC creation

Creating VWOC structures involves the following interfaces:

⎯ VWOCCreator: an interface defining the methods to create VWOC structures.

4.4.4 VWOC access

Accessing VWOC structures involves the following interfaces:

⎯ VWOCParser: an interface defining the methods to parse VWOC structures.

4.5 ISO/IEC 23005-5 APIs

4.5.1 Introduction

This clause specifies the API to the ISO/IEC 23005-5 reference software. The API is defined in Java and follows following package structure:

Name	Definition
`org.iso.mpeg.mxm.engine.iidengine`	Java package name for API to the ISO/IEC 23005-5 reference software.

4.5.2 IID engine

The IIDEngine interface defines the methods for operating over data structures as defined within ISO/IEC 23005-2. Classes implementing the IID engine interface act as factories creating instances of classes performing the following functionalities:

⎯ Classes to create data structures, by means of the IID creation engine

⎯ Classes to access data contained in data structures, by means of IID parser engine

4.5.3 IID creation

Creating IID structures involves the following interfaces:

— IIDCreator: an interface defining the methods to create IID structures.

4.5.4 IID access

Accessing IID structures involves the following interfaces:

— IIDParser: an interface defining the methods to parse IID structures.

4.6 Binary representation APIs for ISO/IEC 23005

4.6.1 Introduction

This clause specifies the API to the ISO/IEC 23005 reference software for binary representation. The API is defined in Java and follows following package structure:

Name	Definition
`org.iso.mpeg.mpegv.binary.core`	Java package name for API to the ISO/IEC 23005 reference software for binary representation.

4.6.2 BinaryIO

The BinaryIO interface defines the methods for operating over data structures as defined within ISO/IEC 23005. Classes implementing the BinaryIO interface act as factories creating instances of classes performing the following functionalities:

— Classes to generate a binary structure from a MXM object tree, by means of the construct binary structure function.

— Classes to generate a binary file from a binary structure, by means of the write binary function.

— Classes to generate a binary structure from a binary structure, by means of the read binary function.

— Classes to restore a MXM object tree from a binary structure, by means of the create MXM object function.

4.6.3 DefaultBinaryIO

DefaultBinaryIO class defines function "Encode" and "Decode". Root elements of other parts (e.g. Part 2, 3, 4) can be extended from this DefaultBinaryIO class. DefaultBinaryIO involves the following functions:

— encode: a function defining the methods to create binary file from a MXM object tree.

— decode: a function defining the methods to restore a MXM object tree from a binary file.

5 Conformance for ISO/IEC 23005

5.1 Introduction

This clause defines conformance for ISO/IEC 23005. A bit-stream is compliant to ISO/IEC 23005 if it conforms to the conformance definitions specified in 5.2 and 5.3.

5.2 Rule-based Conformance for ISO/IEC 23005

5.2.1 Introduction

This clause defines a validation schema based on ISO/IEC 19757-3 to check the conformance bit-streams – XML descriptions – that conform or do not conform to the normative clauses of the other parts of ISO/IEC 23005 and descriptions thereof. In particular, the additional validation rules as specified in the other parts of ISO/IEC 23005 are checked with this validation schema. The actual schema can be found in 5.2.2 and a more readable version of the rules and assertion messages is provided in 5.2.3. The actual conformance bit-streams – XML descriptions – are defined in 5.2.4. The actual conformance bit-streams and corresponding code can be found attached to this document.

For transforming the validation schema according to ISO/IEC 19757-3 and for validating the examples Saxon [1] is used.

5.2.2 Validation schema

```
<schema
  xmlns="http://purl.oclc.org/dsdl/schematron"
  xmlns:sedl="urn:mpeg:mpeg-v:2010:01-SEDL-NS"
  xmlns:sev="urn:mpeg:mpeg-v:2010:01-SEV-NS"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:si="urn:mpeg:mpeg21:2003:01-DIA-XSI-NS"
  xmlns:cidl="urn:mpeg:mpeg-v:2010:01-CIDL-NS"
  xmlns:dcdv="urn:mpeg:mpeg-v:2010:01-DCDV-NS"
  xmlns:scdv="urn:mpeg:mpeg-v:2010:01-SCDV-NS"
  xmlns:sepv="urn:mpeg:mpeg-v:2010:01-SEPV-NS"
  xmlns:mpegvct="urn:mpeg:mpeg-v:2010:01-CT-NS"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
  xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
  xmlns:iidl="urn:mpeg:mpeg-v:2010:01-IIDL-NS"
  xmlns:dcv="urn:mpeg:mpeg-v:2010:01-DCV-NS"
  xmlns:siv="urn:mpeg:mpeg-v:2010:01-SIV-NS"
  queryBinding='xslt' schemaVersion='ISO19757-3'>
  <ns prefix="sedl" uri="urn:mpeg:mpeg-v:2010:01-SEDL-NS"/>
  <ns prefix="si" uri="urn:mpeg:mpeg21:2003:01-DIA-XSI-NS"/>
  <ns prefix="sev" uri="urn:mpeg:mpeg-v:2010:01-SEV-NS"/>
  <ns prefix="xsi" uri="http://www.w3.org/2001/XMLSchema-instance"/>
  <ns prefix="mpeg7" uri="urn:mpeg:mpeg7:schema:2004"/>
  <ns prefix="cidl" uri="urn:mpeg:mpeg-v:2010:01-CIDL-NS"/>
  <ns prefix="dcdv" uri="urn:mpeg:mpeg-v:2010:01-DCDV-NS"/>
  <ns prefix="scdv" uri="urn:mpeg:mpeg-v:2010:01-SCDV-NS"/>
  <ns prefix="sepv" uri="urn:mpeg:mpeg-v:2010:01-SEPV-NS"/>
  <ns prefix="mpegvct" uri="urn:mpeg:mpeg-v:2010:01-CT-NS"/>
  <ns prefix="dia" uri="urn:mpeg:mpeg21:2003:01-DIA-NS"/>
  <ns prefix="iidl" uri="urn:mpeg:mpeg-v:2010:01-IIDL-NS"/>
  <ns prefix="dcv" uri="urn:mpeg:mpeg-v:2010:01-DCV-NS"/>
  <ns prefix="siv" uri="urn:mpeg:mpeg-v:2010:01-SIV-NS"/>
  <title>Schema for validating SEDL/CIDL/IIDL descriptions</title>
  <!-- ################################### -->
  <!-- Schema for validating CIDL(Part 2) descriptions -->
```

```
<!-- ##################################### -->
<pattern name="SensorDeviceCapability element">
  <!-- R1.0: Check the conformance of SensorDeviceCapability -->
  <rule context="cidl:SensorDeviceCapability">
    <assert test="if (@minValue and @maxValue) then if (@minValue > @maxValue)
then false() else true() else true()">
      A minValue shall be less than or equal to a maxValue.
    </assert>
  </rule>
</pattern>
<pattern name="PositionCapability element">
  <!-- R1.1: Check the conformance of PositionCapability -->
  <rule context="scdv:PositionCapability">
    <assert test="if (@minValue and @maxValue) then if (@minValue > @maxValue)
then false() else true() else true()">
      A minValue shall be less than or equal to a maxValue.
    </assert>
  </rule>
</pattern>
<pattern name="OrientationCapability element">
  <!-- R1.2: Check the conformance of OrientationCapability -->
  <rule context="scdv:OrientationCapability">
    <assert test="if (@minValue and @maxValue) then if (@minValue > @maxValue)
then false() else true() else true()">
      A minValue shall be less than or equal to a maxValue.
    </assert>
  </rule>
</pattern>
<pattern name="VelocityCapability element">
  <!-- R1.3: Check the conformance of VelocityCapability -->
  <rule context="scdv:VelocityCapability">
    <assert test="if (@minValue and @maxValue) then if (@minValue > @maxValue)
then false() else true() else true()">
      A minValue shall be less than or equal to a maxValue.
    </assert>
  </rule>
</pattern>
<pattern name="AngularVelocityCapability element">
  <!-- R1.4: Check the conformance of AngularVelocityCapability -->
  <rule context="scdv:AngularVelocityCapability">
    <assert test="if (@minValue and @maxValue) then if (@minValue > @maxValue)
then false() else true() else true()">
      A minValue shall be less than or equal to a maxValue.
    </assert>
  </rule>
</pattern>
<pattern name="AccelerationCapability element">
  <!-- R1.5: Check the conformance of AccelerationCapability -->
  <rule context="scdv:AccelerationCapability">
    <assert test="if (@minValue and @maxValue) then if (@minValue > @maxValue)
then false() else true() else true()">
      A minValue shall be less than or equal to a maxValue.
    </assert>
  </rule>
</pattern>
<pattern name="AngularAccelerationCapability element">
  <!-- R1.6: Check the conformance of AngularAccelerationCapability -->
  <rule context="scdv:AngularAccelerationCapability">
    <assert test="if (@minValue and @maxValue) then if (@minValue > @maxValue)
then false() else true() else true()">
```

```
          A minValue shall be less than or equal to a maxValue.
        </assert>
      </rule>
  </pattern>
  <pattern name="Range element">
    <!-- R2.*: Check the conformance of range -->
    <rule context="scdv:Range">
      <!-- R2.0 -->
      <assert test="if (child::scdv:XminValue > child::scdv:XmaxValue) then
false() else true()">
        An XminValue shall be less than or equal to an XmaxValue.
      </assert>
      <!-- R2.1 -->
      <assert test="if (child::scdv:YminValue > child::scdv:YmaxValue) then
false() else true()">
        An YminValue shall be less than or equal to an YmaxValue.
      </assert>
      <!-- R2.2 -->
      <assert test="if (child::scdv:ZminValue > child::scdv:ZmaxValue) then
false() else true()">
        A ZminValue shall be less than or equal to a ZmaxValue.
      </assert>
    </rule>
  </pattern>
  <pattern name="OrientationRange element">
    <!-- R3.*: Check the conformance of OrientationRange -->
    <rule context="scdv:OrientationRange">
      <!-- R3.0 -->
      <assert test="if (child::scdv:YawMin > child::scdv:YawMax) then false()
else true()">
        An YawMin shall be less than or equal to an YawMax.
      </assert>
      <!-- R3.1 -->
      <assert test="if (child::scdv:PitchMin > child::scdv:PitchMax) then false()
else true()">
        A PitchMin shall be less than or equal to a PitchMax.
      </assert>
      <!-- R3.2 -->
      <assert test="if (child::scdv:RollMin > child::scdv:RollMax) then false()
else true()">
        A RollMin shall be less than or equal to a RollMax.
      </assert>
    </rule>
  </pattern>

  <!-- ################################### -->
  <!-- Schema for validating SEDL(Part 3) descriptions -->
  <!-- ################################### -->
  <pattern name="SEM element">
    <!-- R1.0: Check the conformance of SEM -->
    <rule context="sedl:SEM">
      <assert test="@si:timeScale">
        The SEM element shall have a timeScale attribute.
      </assert>
    </rule>
  </pattern>
  <pattern name="GroupOfEffects elements">
    <!-- R2.*: Check the conformance of GroupOfEffects -->
    <rule context="sedl:GroupOfEffects">
      <!-- R2.0 -->
```