ISO/IEC JTC 1/SC 34

Date: 2006-11-04

L2/06-397

ISO/IEC CD 19757-7

ISO/IEC JTC 1/SC 34/WG 1

Secretariat: Standards Council of Canada

Document Schema Definition Languages (DSDL) — Part 7: Character Repertoire Description Language

Document type: International Standard
Document subtype:
Document stage: (30) Committee
Document language: E

Contents

Page

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.

ISO/IEC 19757-7 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 34, Document Description and Processing Languages.

ISO/IEC 19757 consists of the following parts, under the general title *Document Schema Definition Languages (DSDL)*:

— *Part 1: Overview*

— *Part 2: Regular-grammar-based validation — RELAX NG*

— *Part 3: Rule-based validation — Schematron*

— *Part 4: Namespace-based validation dispatching language — NVDL*

— *Part 5: Datatypes*

— *Part 6: Path-based integrity constraints*

— *Part 7: Character repertoire description language — CRDL*

— *Part 8: Declarative document manipulation*

— *Part 9: Datatype- and namespace-aware DTDs*

— *Part 10: Validation management*

Introduction

This International Standard defines a set of Document Schema Definition Languages (DSDL) that can be used to specify one or more validation processes performed against Extensible Markup Language (XML) documents. A number of validation technologies are standardized in DSDL to complement those already available as standards or from industry.

The main objective of this International Standard is to bring together different validation-related technologies to form a single extensible framework that allows technologies to work in series or in parallel to produce a single or a set of validation results. The extensibility of DSDL accommodates validation technologies not yet designed or specified.

This part of ISO/IEC 19757 provides a language for describing collections of characters defined in ISO/IEC 10646 or Unicode or default grapheme clusters defined in UAX#29. Descriptions in this language may be referenced from schemas. Furthermore, they may also be referenced from forms and stylesheets.

NOTE At present, no schema languages provides mechanisms for referencing to CRDL schemas.

Descriptions of collections need not to be exact. To provide non-exact descriptions, this part of ISO/IEC 19757 provides kernels and hulls, which provide the lower limit and upper limits, respectively.

The structure of this part of ISO/IEC 19757 is as follows. Clause 5 introduces kernels and hulls of collections. Clause 6 shows how Unicode regular expression can be used to describe permissible characters and default grapheme characters. Clause 7 describes the syntax of CRDL schemas. Clause 8 describes the semantics of a correct CRDL schema; the semantics specify when a character is contained by a collection described by a CRDL schema.Clause 9 describes modes of CRDL validators.

Document Schema Definition Languages (DSDL) — Part 7: Character Repertoire Description Language

1 Scope

This part of the International Standard specifies a Character Repertoire Description Language (CRDL). A CRDL schema describes a collection of characters defined in ISO/IEC 10646 or Unicode or default grapheme clusters defined in UAX#29.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies. However, ISO/IEC 10646 references to all versions of ISO/IEC 10646, while Unicode references to all versions of Unicode.

Each of the following documents has a unique identifier that is used to cite the document in the text. The unique identifier consists of the part of the reference up to the first comma.

RELAX NG, *ISO/IEC 19757-2, Document Schema Definition Languages (DSDL) — Part 2: Grammar-based validation — RELAX NG*

W3C XML, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation, 04 February 2004, available at http://www.w3.org/TR/2004/REC-xml-20040204/

W3C XML-Names, *Namespaces in XML*, W3C Recommendation, 14 January 1999, available at http://www.w3.org/TR/1999/REC-xml-names-19990114/

W3C XML Schema Part 2, *XML Schema Part 2: Datatypes Second Edition*, W3C Recommendation, 28 October 2004, available at http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

IETF RFC 3986, *Uniform Resource Identifiers (URI): Generic Syntax*, Internet Standards Track Specification, January 2005, available at http://www.ietf.org/rfc/rfc3987.txt

IETF RFC 3987, *Internationalized Resource Identifiers (IRIs)*, Internet Standards Track Specification, January 2005, available at http://www.ietf.org/rfc/rfc3987.txt

ISO/IEC 10646, *Universal multiple-octet coded Character Set*

Unicode, *The Unicode Standard*

UAX#29, *Text Boundaries*, Unicode Standard Annex #29, 12 October 2006, available at http://www.unicode.org/reports/tr29/

3 Terms and definitions

For the purposes of this part of ISO/IEC 19757, the term "character" as defined in ISO/IEC 10646 or Unicode, the term "default grapheme cluster" as defined in UAX#29, and the following apply.

3.2
collection
a set of characters or default grapheme clusters

3.3
kernel
a set of characters and default grapheme clusters that are guaranteed to be in the collection

3.4
hull
a set of characters and default grapheme clusters that may be in the collection

4 Notation

in(*x*, *A*): character or default grapheme cluster *x* is in collection *A*

notin(*x*, *A*): character or default grapheme cluster *x* is not in collection *A*

unknown(*x*, *A*): it is unknown whether character or default grapheme cluster *x* is in collection *A*

5 Kernel and hull

A kernel contains characters and default grapheme clusters that are guaranteed to be in the collection; the collection may contain other characters or default grapheme clusters. A hull gives an outer boundary so that characters or default grapheme clusters which are not in the hull are guaranteed not to be in the collection; some characters or default grapheme clusters in the hull may not actually be in the collection.

NOTE Because the repertoire of characters in the Universal Character Set is growing, characters may be continuously added to a set. In this case, it is impossible to specify the set exactly, but it is often possible to specify which character is absolutely included, and which character is absolutely excluded.

6 Unicode regular expressions

Unicode regular expressions defined by W3C XML Schema are used for constraining permissible characters or default grapheme clusters. Default grapheme clusters are represented by character strings. For example, one of the representations of Latin small e with acute accent is a base character e (U+0065) followed by an acute accent (U+0301). A Unicode regular expression "e&#x0301" allows this representation.

An expression *e* is an error when strings *u* and *v* match *e* and the first character of *u* occurs in *v* as a non-first character. Conformant implementations of CRDL SHOULD detect this error.

NOTE This error does not happen as long as base characters (the first characters in combining character sequences) do not occur as non-first characters of default grapheme clusters.

7 Syntax

An CRDL schema in the full syntax shall be an XML document valid against the following RELAX NG schema in the compact syntax.

```
default namespace = "toBeSupplied"

start = coll

coll = union | intersection | difference |
      ref | namedCollection | collection

union = element union { commonAtts, coll+ }
intersection = element intersection { commonAtts, coll+ }
difference = element difference { commonAtts, coll+ }
```



```
ref = element ref {commonAtts, attribute href { xsd:anyURI }}
namedCollection = element namedCollection {commonAtts,  attribute ns { xsd:NCName } }

collection =
   element collection {
      commonAtts,
      (text |
         element kernel {commonAtts, text } |
         element hull {commonAtts, text } |
         (element kernel {commonAtts, text }, element hull {commonAtts, text }))
   }

commonAtts =
   attribute minUcsVersion {text}?,
   attribute maxUcsVersion {text}?,
   attribute xml:id {xsd:ID}?
```

The character content of a collection, kernel or hull element is a regular expression as specified in Annex F of W3C XML Schema Part 2.

NOTE The above schema does not allow foreign elements or attributes. The next draft should introduce a NVDL script for allowing such foreign elements or attributes.

8 Semantics

8.1 General

Given a character or grapheme x and a collection *A*, either in(*x*, *A*), notin(*x*, *A*), or unknown(*x*, *A*) holds.

8.2 collection

The semantics of <collection> ... </collection> is defined below.

— Case 1: the content of the collection element is text

It is assumed that this element has a kernel element and a hull element that specifies the content of this collection. The rest is the same as in Case 4.

— Case 2: the collection element has a kernel element but does not have a hull element.

— in(*x*, <collection> ... </collection>) when *x* matches the Unicode regular expression specified as the content of the kernel element.

— notin(*x* <collection> ... </collection>) never holds.

— unknown(*x* <collection> ... </collection>) when in(*x*, <collection> ... </collection>) does not hold.

— Case 3: the given collection element has a hull element but does not have a kernel element.

— in(*x*, <collection> ... </collection>) never holds

— notin(*x* <collection> ... </collection>) when *x* does not match the Unicode regular expression specified as the content of the hull element.

— unknown(*x* <collection> ... </collection>) when notin(*x*, <collection> ... </collection>) does not hold.

— Case 4: the given collection element has a hull element and a kernel element.

  — in(*x*, <collection> ... </collection>) when *x* matches the Unicode regular expression specified as the content of the kernel element.

  — notin(*x* <collection> ... </collection>) when in(*x*, <collection> ... </collection>) does not hold and *x* does not match the Unicode regular expression specified as the content of the hull element.

  — unknown(*x* <collection> ... </collection>) otherwise.

The semantics of Unicode regular expressions depend on the version of the Unicode standard. The author of a CRDL schema may specify the intended versions by specifying the minUcsVersion and maxUcsVersion atributes. If the CRDL processor cannot use some version between these two, it should report an error and may stop normal processing.

When a collection element does not explicitly specify the minUcsVersion attribute, the nearest ancestor element having this attribute is searched. If it is found, its attribute value is used. If not found, there is no lower bound on Unicode versions.

8.3 union

A character or default grapheme cluster is in <union>*A B* </union> if and only if it is in *A* or *B*. It is not in the union if and only if neither it is in *A* nor it is in *B*.

— in(*x*, <union>*A B*</union>) when in(*x*, *A*) or in(*x*, *B*).

— notin(*x*, <union>*A B*</union>) when notin(*x*, *A*) and notin(*x*, *B*).

— unknown(*x*, <union>*A B*</union>) otherwise.

When a union element has one and only one child element, the semantics shall be the same as that of the child element. When a union element has more than two child elements, the semantics shall be the same as that of <union>*A B* </union> where *A* is the first child and *B* is the union of the other child elements.

8.4 intersection

A character or default grapheme cluster is in <intersection>*A B* </intersection> if and only if it is in *A* and *B*. It is not in the intersection if an only if either it is not in *A* or it is not in *B*.

— in(*x*, <intersection>*A B*</intersection>) when in(*x*, *A*) and in(*x*, *B*).

— notin(*x*, <intersection>*A B*</intersection>) when notin(*x*, *A*) or notin(*x*, *B*)

— unknown(*x*, <intersection>*A B*</intersection>) otherwise.

When a intersection element has one and only one child element, the semantics shall be the same as that of the child element. When a intersection element has more than two child elements, the semantics shall be the same as that of <intersection>*A B* </intersection> where *A* is the first child and *B* is the intersection of the other child elements.

8.5 difference

A character or default grapheme cluster is in <difference> *A B* </difference> if and only if it is in *A* and it is not in *B*. It is not in the difference if an only if either it is not in *A* or it is in *B*.

— in(*x*, <difference>*A B*</difference>) when in(*x*, *A*) and notin(*x*, *B*)

— notin(*x*, <difference>*A B*</difference>) when notin(*x*, *A*) or in(*x*, *B*)

— unknown(*x*, <difference>*A B*</difference>) otherwise.

When a difference element has one and only one child element, the semantics shall be the same as that of the child element. When a difference element has more than two child elements, the semantics shall be the same as that of <difference>*A B* </difference> where *A* is the first child and *B* is the union of the other child elements.

8.6 ref

Given <ref href="*uri*"/> , a CRDL schema (say *S*) shall be obtained by deferencing *uri*. When dereferencing *uri* is not successful (e.g., network errors), the CRDL processor should report an error and should continue normal processing by assuming that "unknown" holds. If it is successful, the semantics is defined below:

— in(*x*,<ref href="*uri*"/>) when in(*x*, *S*).

— notin(*x*,<ref href="*uri*"/>) when notin(*x*, *S*).

— unknown(*x*,<ref href="*uri*"/>) when unknown(*x*, *S*).

8.7 namedCollection

<namedCollection name="*ncname*"/> allows the use of named collections. When *ncname* is registered at IANA, the semantics is provided by the registration information. When *ncname* is an implementation-dependent encoding name, the semantics is implementation-dependent.

When the CRDL processor does not support the named collection *ncname*, it should report an error and continue normal processing by assuming that "unknown" holds.

9 Validation

A CRDL processor is a computer program that validates characters and default grapheme clusters against CRDL schemas.

Given a character or default grapheme cluster and a CRDL schema, a CRDL processor shall report "in", "notIn", or "unknown".

Given a Unicode string and a CDRL schema, a CRDL processor first decomposes the string into a sequence of characters and default grapheme clusters, as defined in UAX#29, and examines each of them in sequence. If every character or default grapheme cluster is in the collection, the result is "in". If some character or default grapheme cluster is not in the collection, the result is "notIn". Otherwise, the result is "unknown".

NOTE However, when a CRVL schema is a single collection element, it is possible to compare a given string against a Unicode regular expression without decomposing the string into a sequence of characters or default grapheme clusters. This is done as follows. First, a new Unicode regular expression is created by applying the closure operator (*) to the given Unicode regular expression. Second, the given string is examined against this new Unicode regular expression. Moreover, when a string consisting of a single LF character matches a given Unicode regular expression, it is possible to divide a multi-line string into lines and examine each line independently.

10 Conformance

Different conformant CRDL processors may report different results. There are three reasons for such discrepancies.

The first reason is that dereferencing IRIs may fail. However, the semantics of CRDL is defined so that such failures make conformant CRDL processors err on the safe side. In other words, such failures do not lead to "in" when "notin" or "unknown" would have been reported, and do not lead to "notin" when "in" or "unknown" would have been reported.

The second reason is that the semantics of Unicode regular expressions depends on the Unicode version. Different conformant CRDL processors may behave very differently. For example, one may report "in", while another, "notin".

The third reason is that encodings are implementation dependent. Again, differnt conformant CRDL processors may behave very differently.

Editorial note: Depending on which natural language is used, requirements on CRDL are very different. Some languages use default grapheme clusters heavily, but do not need thousands of characters. CRDL uses Unicode regular expressions for handling default grapheme clusters and further relies on the decomposition of Unicod strings to sequences of characters and default grapheme clusters. Meanwhile, those Asian languages which heavily use ideographic characters need thousands of ideographic characters, which are scattered in the CJK blocks. Such Asian languages do not require default grapheme clusters typically. CRDL provides constructs (union, etc.) for modular representations of huge collections. One could argue that CRDL makes implementations unnecessary difficult, since few people need default grapheme clusters and huge collections at the same time. Comments?

Annex A
(informative)

A.1 8859-6

The charset ISO-8859-6 is described by the following CRDL schema.

```
<collection xmlns="toBeSupplied">[&#x0;- ]|&#xA4;|&#xAD;|&#x60C;|
        &#x61B;|&#x61F;|[&#x621;-&#x63A;]|[&#x640;-&#x652;]</collection>
```

An alternative representation is shown below.

```
<union  xmlns="toBeSupplied">
  <collection>[&#x0;- ]</collection>
  <collection>&#xA4;</collection>
  <collection>&#xAD;</collection>
  <collection>&#x60C;</collection>
  <collection>&#x61B;</collection>
  <collection>&#x61F;</collection>
  <collection>[&#x621;-&#x63A;]</collection>
  <collection>[&#x640;-&#x652;]</collection>
</union>
```

A.2 8859-15

The charset ISO-8859-15 is described by the following CRDL schema.

```
<collection xmlns="toBeSupplied">[&#x0;-&#xA3;] | &#xA5; | &#xA7; |
        [&#xA9;-&#xB3;] | [&#xB5;-&#xB7;] | [&#xB9;-&#xBB;] |
        [&#xBF;-&#xFF;] | [&#x152;-&#x153;] | [&#x160;-&#x161;] |
        &#x178; | [&#x17D;-&#x17E;] | &#x20AC;</collection>
```

A.3 The Japanese list of kanji characters for the first grade

The Japanese Ministry of Education maintains six lists of kanji characters (see Kyouiku kanji[2]). The list for the first grade is described by the following CRDL schema.

```
<union xmlns="toBeSupplied">
  <collection>&#x4E00;</collection>
  <collection>&#x4E03;</collection>
  <collection>[&#x4E09;-&#x4E0B;]</collection>
  <collection>&#x4E2D;</collection>
  <collection>&#x4E5D;</collection>
  <collection>&#x4E8C;</collection>
  <collection>&#x4E94;</collection>
  <collection>&#x4EBA;</collection>
  <collection>&#x4F11;</collection>
  <collection>&#x5148;</collection>
  <collection>&#x5165;</collection>
  <collection>&#x516B;</collection>
  <collection>&#x516D;</collection>
  <collection>&#x5186;</collection>
  <collection>&#x51FA;</collection>
  <collection>&#x529B;</collection>
  <collection>&#x5341;</collection>
  <collection>&#x5343;</collection>
  <collection>&#x53E3;</collection>
  <collection>&#x53F3;</collection>
```

```
    <collection>&#x540D;</collection>
    <collection>&#x56DB;</collection>
    <collection>&#x571F;</collection>
    <collection>&#x5915;</collection>
    <collection>&#x5927;</collection>
    <collection>&#x5929;</collection>
    <collection>&#x5973;</collection>
    <collection>&#x5B50;</collection>
    <collection>&#x5B57;</collection>
    <collection>&#x5B66;</collection>
    <collection>&#x5C0F;</collection>
    <collection>&#x5C71;</collection>
    <collection>&#x5DDD;</collection>
    <collection>&#x5DE6;</collection>
    <collection>&#x5E74;</collection>
    <collection>&#x624B;</collection>
    <collection>&#x6587;</collection>
    <collection>&#x65E5;</collection>
    <collection>&#x65E9;</collection>
    <collection>&#x6708;</collection>
    <collection>&#x6728;</collection>
    <collection>&#x672C;</collection>
    <collection>&#x6751;</collection>
    <collection>&#x6797;</collection>
    <collection>&#x6821;</collection>
    <collection>&#x68EE;</collection>
    <collection>&#x6B63;</collection>
    <collection>&#x6C17;</collection>
    <collection>&#x6C34;</collection>
    <collection>&#x706B;</collection>
    <collection>&#x72AC;</collection>
    <collection>&#x7389;</collection>
    <collection>&#x738B;</collection>
    <collection>&#x751F;</collection>
    <collection>&#x7530;</collection>
    <collection>&#x7537;</collection>
    <collection>&#x753A;</collection>
    <collection>[&#x767D;-&#x767E;]</collection>
    <collection>&#x76EE;</collection>
    <collection>&#x77F3;</collection>
    <collection>&#x7A7A;</collection>
    <collection>&#x7ACB;</collection>
    <collection>&#x7AF9;</collection>
    <collection>&#x7CF8;</collection>
    <collection>&#x8033;</collection>
    <collection>&#x82B1;</collection>
    <collection>&#x8349;</collection>
    <collection>&#x866B;</collection>
    <collection>&#x898B;</collection>
    <collection>&#x8C9D;</collection>
    <collection>&#x8D64;</collection>
    <collection>&#x8DB3;</collection>
    <collection>&#x8ECA;</collection>
    <collection>&#x91D1;</collection>
    <collection>&#x96E8;</collection>
    <collection>&#x9752;</collection>
    <collection>&#x97F3;</collection>
  </union>
```

NOTE One could use a single regular expression, since this list has 80 characters only. However, some other lists of kanji characters have thousands of kanji characters, which prohibit the use of a single regular expression.

A.4 The Japanese list of kanji characters for the second grade

The list for the second grade is described by the following CRDL schema.

```xml
<union xmlns="toBeSupplied">
  <ref href="theURIforTheFirstGrade"/>
  <collection>&#x4E00;</collection>
  <collection>&#x4E03;</collection>
  <collection>[&#x4E09;-&#x4E0B;]</collection>
  <collection>&#x4E2D;</collection>
  <collection>&#x4E5D;</collection>
  <collection>&#x4E8C;</collection>
  <collection>&#x4E94;</collection>
  <collection>&#x4EBA;</collection>
  <collection>&#x4F11;</collection>
  <collection>&#x5148;</collection>
  <collection>&#x5165;</collection>
  <collection>&#x516B;</collection>
  <collection>&#x516D;</collection>
  <collection>&#x5186;</collection>
  <collection>&#x51FA;</collection>
  <collection>&#x529B;</collection>
  <collection>&#x5341;</collection>
  <collection>&#x5343;</collection>
  <collection>&#x53E3;</collection>
  <collection>&#x53F3;</collection>
  <collection>&#x540D;</collection>
  <collection>&#x56DB;</collection>
  <collection>&#x571F;</collection>
  <collection>&#x5915;</collection>
  <collection>&#x5927;</collection>
  <collection>&#x5929;</collection>
  <collection>&#x5973;</collection>
  <collection>&#x5B50;</collection>
  <collection>&#x5B57;</collection>
  <collection>&#x5B66;</collection>
  <collection>&#x5C0F;</collection>
  <collection>&#x5C71;</collection>
  <collection>&#x5DDD;</collection>
  <collection>&#x5DE6;</collection>
  <collection>&#x5E74;</collection>
  <collection>&#x624B;</collection>
  <collection>&#x6587;</collection>
  <collection>&#x65E5;</collection>
  <collection>&#x65E9;</collection>
  <collection>&#x6708;</collection>
  <collection>&#x6728;</collection>
  <collection>&#x672C;</collection>
  <collection>&#x6751;</collection>
  <collection>&#x6797;</collection>
  <collection>&#x6821;</collection>
  <collection>&#x68EE;</collection>
  <collection>&#x6B63;</collection>
  <collection>&#x6C17;</collection>
  <collection>&#x6C34;</collection>
  <collection>&#x706B;</collection>
  <collection>&#x72AC;</collection>
  <collection>&#x7389;</collection>
  <collection>&#x738B;</collection>
  <collection>&#x751F;</collection>
  <collection>&#x7530;</collection>
  <collection>&#x7537;</collection>
  <collection>&#x753A;</collection>
  <collection>[&#x767D;-&#x767E;]</collection>
  <collection>&#x76EE;</collection>
  <collection>&#x77F3;</collection>
  <collection>&#x7A7A;</collection>
  <collection>&#x7ACB;</collection>
  <collection>&#x7AF9;</collection>
  <collection>&#x7CF8;</collection>
```

```
  <collection>&#x8033;</collection>
  <collection>&#x82B1;</collection>
  <collection>&#x8349;</collection>
  <collection>&#x866B;</collection>
  <collection>&#x898B;</collection>
  <collection>&#x8C9D;</collection>
  <collection>&#x8D64;</collection>
  <collection>&#x8DB3;</collection>
  <collection>&#x8ECA;</collection>
  <collection>&#x91D1;</collection>
  <collection>&#x96E8;</collection>
  <collection>&#x9752;</collection>
  <collection>&#x97F3;</collection>
</union>
```


Bibliography

[1] *A Notation for Character Cllections for the WWW*, W3C Note, 14 January 2000, available at http://www.w3.org/TR/charcol

[2] *Kyouiku kanji*, Wikipedia, available at http://en.wikipedia.org/wiki/Ky%C5%8Diku_kanji

