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.

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 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 Information technology — *Document Schema Definition Language (DSDL)*:

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

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

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

— *Part 7: Character repertoire description language (CREPDL)*

— *Part 8: Document schema renaming language (DSRL)*

— *Part 9: Namespace and datatype declaration in Document Type Definitions (DTDs)*

The following parts are under preparation.

— *Part 1: Overview*

— *Part 5: Extensible datatypes*

Introduction

ISO/IEC 19757 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 ISO/IEC 19757 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 character repertoires. Descriptions in this language may be referenced from schemas. Furthermore, they may also be referenced from forms and stylesheets.

NOTE As the time of this writing, no schema languages provide mechanisms for referencing CREPDL schemas.

Descriptions of repertoires need not be exact. Non-exact descriptions are made possible by kernels and hulls, which provide the lower and upper limits, respectively.

The structure of this part of ISO/IEC 19757 is as follows. Clause 5 introduces kernels and hulls of repertoires. Clause 6 specifies the syntax of CREPDL schemas. Clause 7 specifies the semantics of a correct CREPDL schema; the semantics specify when a character is in a repertoire described by a CREPDL schema. Clause 8 defines CREPDL processors and their behaviour. Finally, Annex A describes differences of conformant CREPDL processors, and Annex B provides examples of CREPDL schemas.

Document Schema Definition Languages (DSDL) —

Part 7:
Character Repertoire Description Language (CREPDL)

1 Scope

This part of ISO/IEC 19757 specifies a Character Repertoire Description Language (CREPDL). A CREPDL schema describes a character repertoire.

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.

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.

ISO/IEC 10646, *Information technology — Universal Multiple-Octet Coded Character Set*

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

ISO/IEC 19757-4, *Document Schema Definition Languages (DSDL) — Part 4: Namespace-based validation dispatching language — NVDL*

W3C XML, *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, W3C Recommendation, 16 August 2006, available at http://www.w3.org/TR/2006/REC-xml-20060816

W3C XML-Names, *Namespaces in XML (Second Edition)*, W3C Recommendation, 16 August 2006, available at http://www.w3.org/TR/2006/REC-xml-names-20060816

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

IANA Charsets, *IANA CHARACTER SETS*, available at http://www.iana.org/assignments/character-sets

Unicode, *The Unicode Standard*, The Unicode Consortium, available at http://www.unicode.org/

CLDR, *Unicode Common Locale Data Repository*, The Unicode Consortium, available at http://www.unicode.org/cldr/

3 Terms and definitions

For the purposes of this part of ISO/IEC 19757, the terms "character" and "repertoire" as defined in ISO/IEC 10646 and the following apply.

3.1
kernel
a set of characters that are guaranteed to be in the repertoire

3.2
hull
a set of characters that may be in the repertoire

4 Notation

in(x, A): character x is in the repertoire described by a CREPDL element A

not-in(x, A): character x is not in the repertoire described by a CREPDL element A

unknown(x, A): it is unknown whether character x is in the repertoire described by a CREPDL element A

5 Repertoire, kernel, and hull

A repertoire shall be decribed by specifying a kernel and hull. Kernels and hulls shall be sets of characters.

A character shall be in a repertoire when it is in the kernel. A sequence of characters shall be in a repertoire when any of the characters is in the kernel.

A character shall not be in a repertoire when it is in neither the hull nor the kernel. A sequence of characters shall be not in a repertoire when at least one of the characters is in neither the kernel nor the hull.

It shall be unknown whether or not a character is in a repertoire when it is in the hull but is not in the kernel. It shall be unknown whether or not a sequence of characters is in a reperoire when at least one of the characters is not in the kernel but any of the characters is in the hull or kernel.

NOTE 1 Kernel and hull are borrowed from W3C Note-charcol[3]. Some examples in Annex B also borrowed.

NOTE 2 It may be impossible to specify a repertoire exactly, since characters may continue to be added to the repertoire. However, it is often possible to specify which character is absolutely included, and which character is absolutely excluded. Kernels and hulls help to describe such open repertoires. A kernel is used to specify those characters which are guaranteed to be in the repertoire, while a hull is used to specify an outer boundary. An example of such open repertoires is shown in B.4.

NOTE 3 This part of ISO/IEC 19757 can handle sets of characters, but cannot handle sets of *sequences* of characters. In other words, CREPDL schemas cannot indicate that a combining character is allowed only when it directly follows some base character. Likewise, CREPDL schemas cannot handle named sequences, but can only handle characters occurring in named sequences. It is believed that this part of ISO/IEC 19757 needs this limitation, since implementations become significantly easier.

NOTE 4 It is possible but not recommended to specify a hull that disallows some character in the corresponding kernel. Note that the condition that a character is in a repertoire does not mention the hull.

6 Syntax

6.1 General

An CREPDL schema shall be an XML document (W3C XML) valid against the the NVDL (ISO/IEC 19757-4) script in 6.3, which in turn relies on the RELAX NG (ISO/IEC 19757-2) schema in 6.2. The elements allowed in the RELAX NG schema is of the namespace (W3C XML-Names) `http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0`. Further constraints on the character content of the `char`, `kernel` or `hull` elements are shown in 6.4

NOTE 1 W3C XML 1.1[6] shall not be used for representing CREPDL schemas.

NOTE 2 W3C XML specifies that characters in XML documents are either U+0009 (CHARACTER TABULATION), U+000A (LINE FEED), U+000D (CARRIAGE RETURN), or a character in the ranges from U+0020 to U+D7FF, U+E000 to U+FFFD, or U+10000 to U+10FFFF. In other words, XML documents cannot contain U+0000, U+0001, U+0002, U+0003, U+0004, U+0005, U+0006, U+0007, U+0008, U+000B, U+000C, U+000E, U+000F, U+0010, U+0011, U+0012, U+0013, U+0014, U+0015, U+0016, U+0017, U+0018, U+0019, U+001A, U+001B, U+001C, U+001D, U+001E, or U+001F. Since CREPDL schemas are represented by XML documents, these characters cannot directly occur in CREPDL schemas.

6.2 RELAX NG schema

```
default namespace = "http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0"
```

```
start = coll
coll =
  union | intersection | difference | ref | repertoire | char
union = element union { commonAtts, coll+ }
intersection = element intersection { commonAtts, coll+ }
difference = element difference { commonAtts, coll+ }
ref =
  element ref {
    commonAtts,
    attribute href { xsd:anyURI }
  }
repertoire =
  element repertoire {
    commonAtts,
    attribute registry { text },
    attribute version { text }?,
    (attribute name { text } | attribute number {xsd:int} }
char =
  element char {
    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 }?
# Note that xml:id is allowed, since any foreign attribute is
# allowed by the NVDL script.
```

6.3 NVDL script

```
<?xml version="1.0" encoding="UTF-8"?>
<rules xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
    <namespace ns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
        <validate schema="crepdl.rnc"
                  schemaType="application/relax-ng-compact-syntax">
            <mode>
                <anyNamespace match="elements">
                    <allow/>
                </anyNamespace>
                <namespace ns="" match="attributes">
                    <attach/>
                </namespace>
                <anyNamespace match="attributes">
                    <allow/>
                </anyNamespace>
            </mode>
        </validate>
    </namespace>

</rules>
```

NOTE This NVDL script allows foreign elements and attributes everywhere.

6.4 Regular expressions

The character content of a `char`, `kernel` or `hull` element shall be a regular expression that matches either *Char* or *charClass* as specified in W3C XML Schema Part 2.

NOTE 1 Since this part of ISO/IEC 19757 uses regular expressions for representing sets of characters rather than sets of strings, regular expressions are restricted to *Char* and *charClass*.

NOTE 2 The following rules are duplicated from W3C XML Schema Part 2 for information. The semantics of [29] through [37] depend on the version of Unicode.

```
[10] Char ::= [^.\?*+()|#x5B#x5D]
[11] charClass  ::= charClassEsc | charClassExpr | WildcardEsc
[12] charClassExpr ::= '[' charGroup ']'
[13] charGroup ::= posCharGroup | negCharGroup | charClassSub
[14] posCharGroup ::= ( charRange | charClassEsc )+
[15] negCharGroup ::= '^' posCharGroup
[16] charClassSub ::= ( posCharGroup | negCharGroup )
                     '-' charClassExpr
[17] charRange ::= seRange | XmlCharIncDash
[18] seRange ::= charOrEsc '-' charOrEsc
[20] charOrEsc ::= XmlChar | SingleCharEsc
[21] XmlChar ::= [^\#x2D#x5B#x5D]
[22] XmlCharIncDash ::= [^\#x5B#x5D]
[23] charClassEsc ::= ( SingleCharEsc | MultiCharEsc
                       | catEsc | complEsc )
[24] SingleCharEsc ::= '\' [nrt\|.?*+(){}#x2D#x5B#x5D#x5E]
[25] catEsc ::= '\p{' charProp '}'
[26] complEsc ::= '\P{' charProp '}'
[27] charProp ::= IsCategory | IsBlock
[28] IsCategory ::= Letters | Marks | Numbers
                   | Punctuation | Separators | Symbols | Others
[29] Letters ::= 'L' [ultmo]?
[30] Marks ::= 'M' [nce]?
[31] Numbers ::= 'N' [dlo]?
[32] Punctuation ::= 'P' [cdseifo]?
[33] Separators ::= 'Z' [slp]?
[34] Symbols ::= 'S' [mcko]?
[35] Others ::= 'C' [cfon]?
[36] IsBlock ::= 'Is' [a-zA-Z0-9#x2D]+
[37] MultiCharEsc ::= '\' [sSiIcCdDwW]
[37a] WildcardEsc  ::= '.'
```

NOTE 3 Since W3C REC-xpath-functions[4] extends the definition of regular expressions in W3C XML Schema Part 2, *Char* and *charClass* in W3C REC-xpath-functions[4] and those in W3C XML Schema Part 2 are different in three points. First, *charClass* in W3C REC-xpath-functions[4] allows single character escapes `\^` and `\$`, but that in W3C XML Schema Part 2 does not. Second, *Char* in W3C XML Schema Part 2 allows `$` and `^`, but *Char* in W3C REC-xpath-functions[4] does not. Third, *Char* in W3C XML Schema Part 2 has a known error in which it omits the left brace (`{`) and right brace (`}`), while *Char* in W3C REC-xpath-functions[4] does not omit them.

Implementations of regular expressions in W3C REC-xpath-functions[4] can safely handle the content of a `char`, `kernel` or `hull` element if neither `$` nor `^` appear as *Char* (e.g., `<kernel>$</kernel>`).

7 Semantics

7.1 General

This clause specifies the semantics of a CREPDL element using three notations : in(*x*, *A*), not-in(*x*, *A*), and unknown(*x*, *A*), where *x* is a character and *A* is a CREPDL element. These notations are introduced in Clause 4.

7.2 `char`

First, the semantics of regular expressions occurring in `char`, `kernel`, and `hull` elements shall be as specified in W3C XML Schema Part 2.

NOTE 1 Since regular expressions in W3C XML Schema Part 2 do not satisfy Level-1 conformance requirements in UTS #18[7], implementations of this part of ISO/IEC 19757 do not conform to UTS #18[7].

The semantics of `<char> ... </char>` are defined below.

— Case 1: the `char` element has neither `kernel` nor `hull` as a child element.

It is assumed that this element has a `kernel` element and a `hull` element whose contents are identical to the character content of this element. The rest is the same as in Case 4.

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

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

— not-in(*x*, `<char> ... </char>`) never holds.

— unknown(*x*, `<char> ... </char>`) when *x* does not match the regular expression specified as the content of the `kernel` element.

— Case 3: the `char` element has a `hull` element but does not have a `kernel` element.

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

— not-in(*x*, `<char> ... </char>`) when *x* does not match the regular expression specified as the content of the `hull` element.

— unknown(*x*, `<char> ... </char>`) when *x* matches the regular expression specified as the content of the `hull` element.

— Case 4: the `char` element has a `hull` element and a `kernel` element.

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

— not-in(*x*, `<char> ... </char>`) when *x* does not match the regular expression specified as the content of the `kernel` element and *x* does not match the regular expression specified as the content of the `hull` element.

— unknown(*x*, `<char> ... </char>`) when *x* does not match the regular expression specified as the content of the `kernel` element and *x* matches the regular expression specified as the content of the `hull` element.

Since the semantics of regular expressions depend on the version of the Unicode standard, the author of a CREPDL schema may specify the intended versions by specifying the `minUcsVersion` and `maxUcsVersion` attributes.

EXAMPLE `<char minUcsVersion="4.0" maxUcsVersion="4.0">\p{Nd}</char>` represents the set of characters of the category "Nd" in Unicode Version 4.0.

NOTE 2 It is not guaranteed that every version between these two specify the same properties for every character. However, the author is assumed to accept the discprepancies.

If the CREPDL processor cannot use some version between these two, it should report an error and may stop normal processing.

When a `char` 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. The same applies to `maxUcsVersion`.

7.3 union

First, define the semantics of union elements `<union>`*A B*`</union>`, which contain two child elements *A* and *B*. A character is in the union repertoire described by this element if and only if it is in the one described by *A* or the one described by *B*. It is not in the union repertoire if and only if it is in neither the one described by *A* nor the one described by *B*.

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

— not-in(*x*, `<union>`*A B*`</union>`) when not-in(*x*, *A*) and not-in(*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.

7.4 intersection

First, define the semantics of intersection elements `<intersection>`*A B*`</intersection>`, which contain two child elements *A* and *B*. A character is in the repertoire described by this intersection element if and only if it is in the one described by *A* and it is in the one described by *B*. It is not in this intersection repertore if and only if it is not in the one described by *A* or it is not in the one described by *B*.

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

— not-in(*x*, `<intersection>`*A B*`</intersection>`) when not-in(*x*, *A*) or not-in(*x*, *B*)

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

When an `intersection` element has one and only one child element, the semantics shall be the same as that of the child element. When an `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.

7.5 difference

First, define the semantics of difference elements `<difference>`*A B*`</difference>`, which contain two child elements *A* and *B*. A character is in the repertoire described by this difference element if and only if it is in the one described by *A* and it is not in the one described by *B*. It is not in this difference repertoire if and only if either it is not in the one described by *A* or it is in the one described by *B*.

— in(*x*, `<difference>`*A B*`</difference>`) when in(*x*, *A*) and not-in(*x*, *B*)

— not-in(*x*, `<difference>`*A B*`</difference>`) when not-in(*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.

7.6 ref

Define the semantics of `<ref href="`*iri*`"/>`, where *iri* is an IRI as specified in IETF RFC 3987. First, a CREPDL schema *S* shall be obtained by dereferencing *iri*. When dereferencing *iri* is not successful (e.g., network errors), the CREPDL processor should report an error, and it may stop normal processing or it may continue normal processing by assuming that "unknown" holds. When dereferencing is successful but recursive dereferencing results in an infinite loop, the schema is incorrect. When dereferencing is successful and recursive deferencing does not result in an infintie loop, the semantics are defined below:

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

— not-in(*x*, `<ref href="`*iri*`"/>`) when not-in(*x*, *S*).

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

7.7 repertoire

The `repertoire` element references a repertoire in some registry. The attribute "registry" specifies a registry. The attribute "name" or "number" specifies a repertoire by name or number, respectively.

— When the value of the attribute "registry" is "10646", a collection specified in Annex A of ISO/IEC 10646 is referenced.

— When the value of the attribute "registry" is "CLDR", a locale in the CLDR registry is referenced. When the attribute "version" is present, its value ("1.6", for example) specifies the version of the CLDR registry. For each locale, a POSIX source file, which generated from CLDR, is available. The LC_CTYPE field in the source file lists all characters in the locale. The attribute "number" is ignored.

— When the value of the attribute "registry" is "IANA", a charset in the IANA registry of charsets (IANA Charsets) is referenced. The attribute "name" specifies a name or alias, while the attribute "number" specifies an MIBenum.

— Otherwise, this part of ISO/IEC 19757 does not define the semantics.

The CREPDL processor is not required to recognise repertoires specified by `repertoire` elements. However, when the CREPDL processor does not recognise the specified repertoire, it should report an error. It may continue normal processing by assuming that "unknown" holds or it may stop normal processing.

Even when the repertoire specified by a `repertoire` element is recognised, different CREPDL processors may report different results.

8 Validation

A CREPDL processor is a computer program that validates characters against CREPDL schemas.

When a CREPDL schema is incorrect, a CREPDL processor should report errors and halt.

Given a character and a correct CREPDL schema , a CREPDL processor shall report "in", "not-in", or "unknown".

Given a string and a correct CREPDL schema, a CREPDL processor shall first decompose the string into a sequence of characters, examine each of them in sequence, and report "in", "not-in", or "unknown".

Annex A
(informative)

Differences of Conformant Processors

Different conformant CREPDL processors may report different results only in the cases shown below:

- Case 1: Dereferencing IRIs may fail. However, the semantics of CREPDL is defined so that such failures make conformant CREPDL processors err on the safe side. In other words, such failures do not lead to "in" when "not-in" or "unknown" would have been reported, and do not lead to "not-in" when "in" or "unknown" would have been reported.

- Case 2: The semantics of regular expressions depends on the Unicode version. Different conformant CREPDL processors may behave very differently. For example, one may report "in", while another, "not-in".

- Case 3: A repertoire specified by a `repertoire` element may be unrecognised by the CREPDL processor. Moreover, even when the repertoire is recognised, different CREPDL processors may have different interpretations of the repertoire.

- Case 4: Before CREPDL processors receive CREPDL schemas as well as characters orstrings, character normalization (UAX #15[8]) may be applied. Such character normalization may cause two conformant CREPDL processors to behave very differently.

Annex B
(informative)

Example CREPDL schemas

B.1 ISO/IEC 8859-6

The repertoire of ISO/IEC 8859-6[1] is described by the following CREPDL schema.

```
<union xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
  <char>\p{IsBasicLatin}</char>
  <char
>[&#xA0;&#xA4;&#xAD;&#x60C;&#x61B;&#x61F;&#x621;-&#x63A;&#x640;-&#x652;]</char>
</union>
```

An alternative schema is shown below.

```
<union xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
  <char>\p{IsBasicLatin}</char>
  <char>&#xA0;</char>
  <char>&#xA4;</char>
  <char>&#xAD;</char>
  <char>&#x60C;</char>
  <char>&#x61B;</char>
  <char>&#x61F;</char>
  <char>[&#x621;-&#x63A;]</char>
  <char>[&#x640;-&#x652;]</char>
</union>
```

Yet another alternative is to rely on the IANA registry.

```
<repertoire  xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0"
  registry="IANA" name="ISO_8859-6:1987" />
```

B.2 ISO/IEC 8859-15

The repertoire of ISO/IEC 8859-15[2] is described by the following CREPDL schema.

```
<union xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
  <char>\p{IsBasicLatin}</char>
  <char>[&#xA0;-&#xA3;]</char>
  <char>&#xA5;</char>
  <char>&#xA7;</char>
  <char>[&#xA9;-&#xB3;]</char>
  <char>[&#xB5;-&#xB7;]</char>
  <char>[&#xB9;-&#xBB;]</char>
  <char>[&#xBF;-&#xFF;]</char>
  <char>[&#x152;-&#x153;]</char>
  <char>[&#x160;-&#x161;]</char>
  <char>&#x178;</char>
  <char>[&#x17D;-&#x17E;]</char>
```

```
    <char>&#x20AC;</char>
  </union>
```

An alternative is to rely on the IANA registry. The value of the attribute `"number"` is 111, which is the MIBenum of the charset ISO-8859-15.

```
  <repertoire  xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0"
    registry="IANA" number="111" />
```

B.3 Armenian script

Collection 11 "ARMENIAN" is shown in Appendix A of ISO/IEC 10646. This collection is fixed. In other words, every character in this collection is assiged already and no characters will be assigned later.

A CREPDL schema for referencing this collection is shown below. `number="11"` can be replaced with `name="ARMENIAN"`.

```
  <repertoire  xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0"
    registry="10646" number="11"/>
```

An alternative is to use a regular expression that specifies the characters explicitly.

```
  <char xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0"
  >[&#x0530;-&#x058F;]</char>
```

B.4 Malayalam script

Collection 24 "MALAYALAM" is shown in Appendix A of ISO/IEC 10646. Unlike in the previous example, this collection is not fixed but open. In other words, this collection has some unassinged characters, which may be assinged later.

A CREPDL schema for referencing this collection is shown below. Again, `number="24"` can be replaced with `name="MALAYALAM"`.

```
  <repertoire  xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0"
    registry="10646" number="24"/>
```

An alternative is to use regular expressions that specifies the characters explicitly. Although it is possible to use a single regular expression, the CREPDL schema shown below uses two regular expressions and wrap them with a `union` element.

```
  <union xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
    <char>[&#x0D00;-&#x0D7F;]</char>
    <char>[&#x200C;-&#x200d;]</char>
  </union>
```

However, this CREPDL schema allows unassigned characters such as U+0D11. A better solution is create a hull containing the characters from U+0D00 to U+0D7F, U+200C, and U+200D and a kernel containing the assigned characters only.

```
  <char xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
      <kernel>[...assinged characters...]</kernel>
```

```
    <hull>[&#x0D00;-&#x0D7F;&#x200C;-&#x200d;]</hull>
</char>
```

The kernel element in this example is omitted for readability. Since this single element describes all assigned characters in this collection, it is lengthy and almost unreadable. Use of union and intersection elements leads to a more readable CREPDL schema.

```
<intersection  xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
  <union>
    <char><kernel>[&#xD02;-&#xD03;]</kernel></char>
    <char><kernel>[&#xD05;-&#xD0C;]</kernel></char>
    <char><kernel>[&#xD0E;-&#xD10;]</kernel></char>
    <char><kernel>[&#xD12;-&#xD28;]</kernel></char>
    <char><kernel>[&#xD2A;-&#xD39;]</kernel></char>
    <char><kernel>[&#xD3D;-&#xD44;]</kernel></char>
    <char><kernel>[&#xD46;-&#xD48;]</kernel></char>
    <char><kernel>[&#xD4A;-&#xD4D;]</kernel></char>
    <char><kernel>&#xD57;</kernel></char>
    <char><kernel>[&#xD60;-&#xD63;]</kernel></char>
    <char><kernel>[&#xD66;-&#xD75;]</kernel></char>
    <char><kernel>[&#xD79;-&#xD7F;]</kernel></char>
  </union>
  <char>[&#x0D00;-&#x0D7F;&#x200C;-&#x200d;]</char>
</intersection>
```

An alternative is to use union only. This alternative is more concise and probably easier to understand.

```
<union  xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
  <union>
    <char>[&#xD02;-&#xD03;]</char>
    <char>[&#xD05;-&#xD0C;]</char>
    <char>[&#xD0E;-&#xD10;]</char>
    <char>[&#xD12;-&#xD28;]</char>
    <char>[&#xD2A;-&#xD39;]</char>
    <char>[&#xD3D;-&#xD44;]</char>
    <char>[&#xD46;-&#xD48;]</char>
    <char>[&#xD4A;-&#xD4D;]</char>
    <char>&#xD57;</char>
    <char>[&#xD60;-&#xD63;]</char>
    <char>[&#xD66;-&#xD75;]</char>
    <char>[&#xD79;-&#xD7F;]</char>
  </union>
  <char><hull>[&#x0D00;-&#x0D7F;&#x200C;-&#x200d;]</hull></char>
</union>
```

B.5 The Japanese list of kanji characters for the first grade

The Japanese Ministry of Education, Culture, Sports, Science and Technology maintains six lists of kanji characters (see Gakunenbetsu kanji haitouhyou[5]). The list for the first grade is described by the following CREPDL schema, and this list contains 80 characters.

```
<union xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
  <char>[&#x4e00;&#x53f3;&#x96e8;&#x5186;&#x738b;]</char>
  <char>[&#x97f3;&#x4e0b;&#x706b;&#x82b1;&#x8c9d;]</char>
  <char>[&#x5b66;&#x6c17;&#x4e5d;&#x4f11;&#x7389;]</char>
  <char>[&#x91d1;&#x7a7a;&#x6708;&#x72ac;&#x898b;]</char>
  <char>[&#x53e3;&#x6821;&#x5de6;&#x4e09;&#x5c71;]</char>
```

```
    <char>[&#x5b50;&#x56db;&#x7cf8;&#x5b57;&#x8033;]</char>
    <char>[&#x4e03;&#x8eca;&#x624b;&#x5341;&#x51fa;]</char>
    <char>[&#x5973;&#x5c0f;&#x4e0a;&#x68ee;&#x4eba;]</char>
    <char>[&#x6c34;&#x6b63;&#x751f;&#x9752;&#x5915;]</char>
    <char>[&#x77f3;&#x8d64;&#x5343;&#x5ddd;&#x5148;]</char>
    <char>[&#x65e9;&#x8349;&#x8db3;&#x6751;&#x5927;]</char>
    <char>[&#x7537;&#x7af9;&#x4e2d;&#x866b;&#x753a;]</char>
    <char>[&#x5929;&#x7530;&#x571f;&#x4e8c;&#x65e5;]</char>
    <char>[&#x5165;&#x5e74;&#x767d;&#x516b;&#x767e;]</char>
    <char>[&#x6587;&#x6728;&#x672c;&#x540d;&#x76ee;]</char>
    <char>[&#x7acb;&#x529b;&#x6797;&#x516d;&#x4e94;]</char>
  </union>
```

NOTE One could use a single regular expression. However, some other lists of kanji characters have thousands of kanji characters, which prohibit the use of a single regular expression.

B.6 The Japanese list of kanji characters for the second grade

The list for the second grade is described by the following CREPDL schema. It contains 160 characters.

```
  <union xmlns="http://purl.oclc.org/dsdl/crepdl/ns/structure/1.0">
  <union>
    <char>[&#x5f15;&#x7fbd;&#x96f2;&#x5712;&#x9060;]</char>
    <char>[&#x4f55;&#x79d1;&#x590f;&#x5bb6;&#x6b4c;]</char>
    <char>[&#x753b;&#x56de;&#x4f1a;&#x6d77;&#x7d75;]</char>
    <char>[&#x5916;&#x89d2;&#x697d;&#x6d3b;&#x9593;]</char>
    <char>[&#x4e38;&#x5ca9;&#x9854;&#x6c7d;&#x8a18;]</char>
    <char>[&#x5e30;&#x5f13;&#x725b;&#x9b5a;&#x4eac;]</char>
    <char>[&#x5f37;&#x6559;&#x8fd1;&#x5144;&#x5f62;]</char>
    <char>[&#x8a08;&#x5143;&#x8a00;&#x539f;&#x6238;]</char>
    <char>[&#x53e4;&#x5348;&#x5f8c;&#x8a9e;&#x5de5;]</char>
    <char>[&#x516c;&#x5e83;&#x4ea4;&#x5149;&#x8003;]</char>
    <char>[&#x884c;&#x9ad8;&#x9ec4;&#x5408;&#x8c37;]</char>
    <char>[&#x56fd;&#x9ed2;&#x4eca;&#x624d;&#x7d30;]</char>
    <char>[&#x4f5c;&#x7b97;&#x6b62;&#x5e02;&#x77e2;]</char>
    <char>[&#x59c9;&#x601d;&#x7d19;&#x5bfa;&#x81ea;]</char>
    <char>[&#x6642;&#x5ba4;&#x793e;&#x5f31;&#x9996;]</char>
    <char>[&#x79cb;&#x9031;&#x6625;&#x66f8;&#x5c11;]</char>
    <char>[&#x5834;&#x8272;&#x98df;&#x5fc3;&#x65b0;]</char>
    <char>[&#x89aa;&#x56f3;&#x6570;&#x897f;&#x58f0;]</char>
    <char>[&#x661f;&#x6674;&#x5207;&#x96ea;&#x8239;]</char>
    <char>[&#x7dda;&#x524d;&#x7d44;&#x8d70;&#x591a;]</char>
    <char>[&#x592a;&#x4f53;&#x53f0;&#x5730;&#x6c60;]</char>
    <char>[&#x77e5;&#x8336;&#x663c;&#x9577;&#x9ce5;]</char>
    <char>[&#x671d;&#x76f4;&#x901a;&#x5f1f;&#x5e97;]</char>
    <char>[&#x70b9;&#x96fb;&#x5200;&#x51ac;&#x5f53;]</char>
    <char>[&#x6771;&#x7b54;&#x982d;&#x540c;&#x9053;]</char>
    <char>[&#x8aad;&#x5185;&#x5357;&#x8089;&#x99ac;]</char>
    <char>[&#x58f2;&#x8cb7;&#x9ea6;&#x534a;&#x756a;]</char>
    <char>[&#x7236;&#x98a8;&#x5206;&#x805e;&#x7c73;]</char>
    <char>[&#x6b69;&#x6bcd;&#x65b9;&#x5317;&#x6bce;]</char>
    <char>[&#x59b9;&#x4e07;&#x660e;&#x9cf4;&#x6bdb;]</char>
    <char>[&#x9580;&#x591c;&#x91ce;&#x53cb;&#x7528;]</char>
    <char>[&#x66dc;&#x6765;&#x91cc;&#x7406;&#x8a71;]</char>
  </union>
```

Bibliography

[1] *ISO/IEC 8859-6, Information technology — 8-bit single-byte coded graphic character sets — Part 6: Latin/Arabic alphabet*

[2] *ISO/IEC 8859-15, Information technology — 8-bit single-byte coded graphic character sets, — Part 15: Latin alphabet No. 9*

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

[4] *XQuery 1.0 and XPath 2.0 Functions and Operators*, W3C Recommendation, 23 January 2007, available at http://www.w3.org/TR/2007/REC-xpath-functions-20070123/

[5] *Gakunenbetsu kanji haitouhyou (in Japanese)*, 14 December 1998, available at http://www.mext.go.jp/b_menu/shuppan/sonota/990301b/990301d.htm

[6] *Extensible Markup Language (XML) 1.1 (Second Edition)*, W3C Recommendation, 16 August 2006, available at http://www.w3.org/TR/2006/REC-xml11-20060816

[7] *Unicode Regular Expressions*, Unicode Technical Standard #18, available at http://unicode.org/reports/tr18/

[8] *Unicode Normalization Forms*, Unicode Standard Annex #15, available at http://unicode.org/reports/tr15/