W3C Editor's Draft

rdf:text: A Datatype for Internationalized Text

W3C Editor's Draft 28 November 2008

This version:
http://www.w3.org/2007/OWL/draft/ED-owl2-rdf-text-20081128/
Latest editor's draft:
http://www.w3.org/2007/OWL/draft/owl2-rdf-text/
Previous version:
http://www.w3.org/2007/OWL/draft/ED-owl2-rdf-text-20081126/ (color-coded diff)
Authors:
Jie Bao, Rensselaer Polytechnic Institute, Troy, New York, USA
Axel Polleres, DERI Galway at the National University of Ireland, Galway, Ireland
Boris Motik, Oxford University, Oxford, UK

This document is also available in these non-normative formats: PDF version.

Abstract

This document presents the specification for a primitive datatype representing internationalized text that is used in both the RIF and OWL languages.

Status of this Document

May Be Superseded

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

W3C Editor's Draft

Set of Documents

This document is being published as one of a set of 11 documents:

1. Structural Specification and Functional-Style Syntax
2. Direct Semantics
3. RDF-Based Semantics
4. Conformance and Test Cases
5. Mapping to RDF Graphs
6. XML Serialization
7. Profiles
8. Quick Reference Guide
9. New Features and Rationale
10. Manchester Syntax
11. rdf:text: A Datatype for Internationalized Text (this document)

Please Comment By 2008-12-01

The OWL Working Group seeks public feedback on these Working Drafts. Please send your comments to public-owl-comments@w3.org (public archive). If possible, please offer specific changes to the text that would address your concern. You may also wish to check the Wiki Version of this document for internal-review comments and changes being drafted which may address your concerns.

No Endorsement

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

Patents

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Contents

W3C Editor's Draft

1 Introduction

Internationalized text — that is, text that additionally conveys information in terms of a language tag — is used in several existing W3C specifications, such as RDF, XML, OWL, and RIF. This specification defines a datatype called `rdf:text` in order to allow specifications such as RDF, OWL, and RIF to refer to internationalized text literals in an interoperable way. Parallel efforts have been made to support internationalized strings by several W3C working groups, including the OWL WG and the RIF WG. Collaboration between the two working groups on the choice of language constructs for internationalized strings has lead to the present specification [1][2].

Parts of this document are based on the current work on rif:text [3] (RIF WG) and owl:internationalizedString [4] (OWL WG), for more details see a summary.

2 Preliminaries

A *character* is an atomic unit of communication. The structure of characters is not further specified in this document, other than to note that each character has a Universal Character Set (UCS) code point [*ISO/IEC 10646*] (or, equivalently, a Unicode code point [*UNICODE*]). The set of available characters is assumed to be infinite, and it is thus independent from the current version of UCS and Unicode.

A *string* is a finite sequence of characters. The *length* of a string is the number of characters in it. Strings are written in this specification by enclosing them in quotes. Two strings are identical if they contain exactly the same sequence of characters.

Example:

To understand the rationale behind the assumption on the infinite number of characters, consider the following OWL 2 ontology:

```
ClassAssertion( a:i MinCardinality( n a:some-property
DatatypeRestriction( xs:string xs:length 1 ) ) )
```

Intuitively, this OWL 2 axiom states that the individual `a:i` is connected to at least *n* different strings of length 1. If one assumes that there are exactly *m* UCS characters, then this ontology is satisfiable if and only if $n \leq m$. This has several undesirable consequences:

- OWL 2 reasoners need to know exactly how many UCS characters there are.
- Changing the number of UCS characters might change satisfiability of an ontology.

In order to avoid such problems, this specification assumes that the number of UCS characters is infinite; that is, $m = \infty$. Despite this assumption, at any given point in time, UCS provides means of addressing only a finite subset of this set.

Thus, the example ontology is satisfiable regardless of with respect to which version of UCS it is interpreted.

A *language tag* is a string of the form as specified in BCP 47 [*BCP 47*].

This specification uses Uniform Resource Identifiers (URIs) for naming datatypes and their components, which are defined in RFC 3986 [*RFC 3986*]. For readability, URIs are often abbreviated according to the convention of XML Namespaces [*XML Namespaces*]. The following namespace prefixes are used throughout this document:

- the `xs:` prefix stands for the XML Schema namespace URI `http://www.w3.org/2001/XMLSchema#`
- the `rdf:` prefix stands for `http://www.w3.org/1999/02/22-rdf-syntax-ns#`

Datatypes are defined in this document along the lines of XML Schema Datatypes [*XML Schema Datatypes*]. Each datatype is identified by a URI and is described by the following components:

- The *value space* is a set determining the set of values of the datatype. Elements of the value space are called *data values*.
- The *lexical space* is a set of strings that can be used to refer to data values. Each member of the lexical space is called a *lexical value*, and it is mapped to a particular data value.

- The *facet space* is a set of pairs of the form ⟨ F v ⟩, where F is a URI called a *constraining facet*, and v is an arbitrary object called a *value*. Each such pair is mapped to a subset of the value space of the datatype.

The italicized keywords *must*, *must not*, *should*, *should not*, and *may* specify certain aspects of the normative behavior of tools implementing this specification, and are interpreted as specified in RFC 2119 [*RFC 2119*].

3 Definition of the rdf:text Datatype

The datatype identified by the URI http://www.w3.org/1999/02/22-rdf-syntax-ns#text (abbreviated rdf:text) allows for the representation of internationalized text strings. In addition to the RIF and OWL specifications, this datatype is expected to supersede RDF's plain literals with language tags, cf. [5], which is why this datatype has been added into the rdf: namespace.

Value Space. The value space of rdf:text is the set of all pairs of the form ("*text*" , "*lang*"), where "*text*" is a string and "*lang*" is either the empty string "" or a lowercase language tag.

Lexical Space. A lexical value of rdf:text is a string "val" that contains at least one @ character (U+40) and that satisfies the following condition:

> Let *i* be the position of the last @ (U+40) character in "val", and let "abc" and "tag" be the substrings of "val" containing the characters up to and after position *i* (noninclusive), respectively. Then ,"tag" *must* be either empty or a valid language tag.

Each such lexical value is assigned a data value ("abc", "lc-tag"), where "lc-tag" is the string "tag" converted to lowercase.

Editor's Note: Open Issues: The definition of the set of characters, particularly the fact that it is infinite, as well as the compatibility with XML strings - whether the string part of the lex & val space should be the same as xs:string - are still under discussion.

Example:

Lexical value "Family Guy@en" is mapped to the data value ("Family Guy" , "en"), and "Family Guy@" is mapped to ("Family Guy" , ""). Furthermore, "Family Guy" is not a valid lexical value of rdf:text because it does not contain the @ (U+40) character.

Facet Space. The facet space of the rdf:text datatype is shown in Table 1.

Table 1. The Facet Space of the `rdf:text` Datatype

A pair of the form...	**...is mapped to the subset of the value space of `rdf:text` containing all pairs of the form ("*text*" , "*lang*") such that...**
〈 `xs:minLength v` 〉 where `v` is a nonnegative integer	the length of "*text*" is at least `v`
〈 `xs:maxLength v` 〉 where `v` is a nonnegative integer	the length of "*text*" is at most `v`
〈 `xs:length v` 〉 where `v` is a nonnegative integer	the length of "*text*" is exactly `v`
〈 `xs:pattern v` 〉 where `v` is a string specifying a regular expression with the syntax as in Section F of XML Schema Datatypes [*XML Schema Datatypes*]	"*text*" matches the regular expression `v`
〈 `rdf:langPattern v` 〉 where `v` is a string specifying a regular expression with the syntax as in Section F of XML Schema Datatypes [*XML Schema Datatypes*]	"*lang*" matches the regular expression `v`

Editor's Note: TODO: an example of pattern and langPattern, e.g., a language pattern that matches all English dialects including en-US and en-UK

3.1 `xs:string` as a restriction of `rdf:text`

The `xs:string` datatype is a datatype defined in XML Schema Datatypes [*XML Schema Datatypes*] as having the value space equal to the set of all strings. Thus, the value space of `xs:string` is not a subset of the value space of `rdf:text`, which may cause problems for certain applications of this specification. A similar problem arises with XML Schema datatypes that are derived from `xs:string`.

To overcome this difficulty, specifications that use `rdf:text` *may* choose to interpret the datatypes from the following list in a slightly different way. The resulting datatypes have value spaces that are isomorphic with the value spaces from XML Schema Datatypes [*XML Schema Datatypes*], but that are subsets of the value space of `rdf:text`.

- `xs:string`
- `xs:normalizedString`
- `xs:token`
- `xs:language`
- `xs:Name`
- `xs:NCName`
- `xs:NMTOKEN`

Value Space. For *DT* a datatype from the above list, the value space of *DT* is a set of pairs of the form `( "text" , "" )` where `"text"` is a string matching the restrictions of *DT* as specified in XML Schema Datatypes [*XML Schema Datatypes*] and `""` is the empty string.

Lexical Space. For *DT* a datatype from the above list, the lexical space of *DT* is a string `"text"` that matches the restrictions of *DT* as specified in XML Schema Datatypes [*XML Schema Datatypes*]. Each lexical value `"text"` is assigned a data value `( "text" , "" )`.

Facet Space. Each datatype *DT* from the above list supports the constraining facets `xs:minLength`, `xs:maxLength`, `xs:length`, and `xs:pattern`. The facet value of each pair for *DT* is the same as in Table 1, with the difference that the result is a subset of the value space of *DT* rather than of `rdf:text`.

3.2 Abbreviations of `rdf:text` and `xs:string` Literals

In syntaxes such as the RIF presentation syntax [6], the OWL 2 functional-style syntax [7], or the TURTLE syntax [8], literals are written using the form `"rep"^^datatypeURI`. This specification defines a convenient representation for `rdf:text` and `xs:string` literals. In particular, literals of the form `"text@lang"^^rdf:text` where `"lang"` is not empty can be abbreviated as `"text"@lang`; furthermore, literals of the form `"text"^^xs:string` can be abbreviated as `"text"`. If an implementation supports abbreviation of literals, it *should* abbreviate the literals eagerly whenever possible.

The abbreviated literals can be written using the following grammar. A subset of the N-triples quoting mechanism is employed in order to allow strings to contain quotes.

```
quotedString := 'a finite sequence of characters in which "
(U+22) and \ (U+5C) occur only in pairs of the form \"
(U+22, U+5C) and \\ (U+22, U+22), enclosed in a pair of "
(U+22) characters'
languageTag := a nonempty (not quoted) string defined as
specified in BCP-47 [BCP-47]
abbreviatedXSDStringLiteral := quotedString
abbreviatedRDFTextLiteral := quotedString '@' languageTag
abbreviatedLiteral := abbreviatedXSDStringLiteral |
abbreviatedRDFTextLiteral
```

Text matching the **abbreviatedXSDStringLiteral** production *should* be mapped to an `xs:string` literal, and text matching the **abbreviatedRDFTextLiteral** production *should* be mapped to an `rdf:text` literal.

Example:

`"Padre de familia"@es` is an abbreviation for the `rdf:text` literal `"Padre de familia@es"^^rdf:text` — a literal denoting a pair consisting of the string `"Padre de familia"` and the language tag `es` denoting the Spanish language. Furthermore, `"Padre de familia"` is an abbreviation for an `xs:string` literal `"Padre de familia"^^xs:string`, which is mapped to the same data value as the `rdf:text` literal `"Padre de familia@"^^rdf:text`.

4 Functions and Operators on `rdf:text`

This section defines constructor functions, operators, and functions on the `rdf:text` datatype. The terminology used and structure to describe these functions and operators is in accordance with the XQuery 1.0 and XPath 2.0 Functions and Operators [*XPathFunc*]. The error codes used in this section are given in Appendix G of the XPath 2.0 specification [*XPath20*] and Appendix C of XQuery and XPath function specification [*XPathFunc*].

Editor's Note: Reuse of the `fn:` namespace in the following functions is still under discussion, cf. http://lists.w3.org/Archives/Public/public-rdf-text/2008OctDec/0020.html

4.1 Functions for Assembling and Disassembling of `rdf:text` Literals

4.1.1 fn:text-from-string-lang

```
fn:text-from-string-lang( $arg1 as xs:string, $arg2 as xs:string) as rdf:te
```

Summary: returns the data value `( $arg1, $arg2 )` of type `rdf:text`. The arguments both have to be of type `xs:string` or one of its subtypes additionally, $arg2 has to be valid language tag according to BCP-47 [*BCP-47*]; otherwise, this function raises type error err:FORG0006.

4.1.2 fn:text-from-string

```
fn:text-from-string( $arg as xs:string) as rdf:text
```

Summary: returns the data value `($arg, "")` of type `rdf:text`. The argument has to be of type `xs:string` or one of its subtypes; otherwise, this function raises type error err:FORG0006.

4.1.3 fn:string-from-text

```
fn:string-from-text( $arg as rdf:text) as xs:string
```

Summary: extracts the string part `s` from the argument `$arg = ( s, l )` of type `rdf:text`. The argument `$arg` has to be of type `rdf:text`; otherwise, this function raises type error err:FORG0006.

4.1.4 fn:lang-from-text

```
fn:lang-from-text( $arg as rdf:text ) as xs:lang
```

Summary: extracts the language tag `l` from the argument `$arg = ( s, l )` of type `rdf:text`. The argument `$arg` has to be of type `rdf:text`; otherwise, this function raises type error err:FORG0006.

4.2 Comparison of `rdf:text` Values

The notion of collations used in this section is taken from Section 7.3.1 of XPath and XQuery function specification [*XPathFunc*].

4.2.1 fn:text-compare

```
fn:text-compare( $comparand1  as rdf:text?, $comparand2 as rdf:text? ) as x

fn:text-compare( $comparand1  as rdf:text?, $comparand2 as rdf:text?, $coll
```

Summary: returns the empty sequence if one of the arguments is empty or if the language parts of `$comparand1` and `$comparand2` are unequal; otherwise, this function returns -1, 0, or 1 depending on whether the value of the string-part of `$comparand1` is respectively less than, equal to, or greater than the value of the string-part of `$comparand2`. The collation used by the invocation of this function is determined according to the rules in Section 7.3.1 of the XPath and XQuery functions specification [*XPathFunc*].

This function, invoked with the first signature, backs up the "eq", "ne", "gt", "lt", "le" and "ge" operators on text values.

The two functions may be viewed as declared XQuery functions with the following definitions:

```
declare function fn:text-compare( $comparand1 as rdf:text?, $comparand2 as
 {
  return
    if ( fn:compare ( fn:lang-from-text( $comparand1 ),  fn:lang-from-text(
```

```
        fn:compare ( fn:string-from-text( $comparand1 ) , fn:string-from-tex
 }


declare function fn:text-compare( $comparand1  as rdf:text?, $comparand2 as
 {
  return
    if ( fn:compare ( fn:lang-from-text( $comparand1 ),  fn:lang-from-text(
       fn:compare ( textstring-from-text( $comparand1 ) , textstring-from-t
 }
```

4.3 Other Functions on `rdf:text` Values

Editor's Note: Open Issues: The inclusion of text-length, as well as the definition of the function - whether the length of an rdf:text value should concern only the string part - are still under discussion.

4.3.1 fn:text-length

```
fn:text-length($arg as rdf:text) as xs:integer
```

Summary: returns the number of characters that constitute the string part of `$arg`.

This function may be viewed as a declared XQuery function with the following definition:

```
declare function fn:text-length($arg as rdf:text?) as xs:integer
 {
  return
     fn:string-length ( fn:string-from-text( $arg ) )
 }
```

4.3.2 fn:matches-language-range

```
fn:matches-language-range($input as rdf:text?, $range as xs:string) as xs:b
```

Summary: returns `true` if the language tag part of `$input` is a valid language tag according to BCP-47 [*BCP-47*], and if it matches the language-range expression supplied as `$range` as specified by the algorithm for "Matching of Language Tags" which is part of BCP-47 [*BCP-47*]; otherwise, it returns `false`.

An empty input sequence is treated as a `rdf:text` value consisting of the empty string and the empty language tag. Since the empty string is not a valid language tag according to BCP-47 [*BCP-47*], on such input this function returns `false`.

5 References

[RFC 3986]
RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax. T. Berners-Lee, R. Fielding, and L. Masinter, IETF, January 2005.

[RFC 2119]
RFC 2119: Key words for use in RFCs to Indicate Requirement Levels. Network Working Group, S. Bradner. Internet Best Current Practice, March 1997.

[UNICODE]
The Unicode Standard. The Unicode Consortium.

[ISO/IEC 10646]
ISO/IEC 10646-1:2000. Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane and ISO/IEC 10646-2:2001. Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 2: Supplementary Planes, as, from time to time, amended, replaced by a new edition or expanded by the addition of new parts. [Geneva]: International Organization for Standardization. ISO (International Organization for Standardization).

[BCP 47]
BCP-47 - Tags for Identifying Languages. A. Phillips, M. Davis, eds., IETF, September 2006, http://www.rfc-editor.org/rfc/bcp/bcp47.txt.

[XML Namespaces]
Namespaces in XML 1.0 (Second Edition). Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin, eds. W3C Recommendation 16 August 2006.

[XML Schema Datatypes]
XML Schema Part 2: Datatypes Second Edition. Paul V. Biron and Ashok Malhotra, eds. W3C Recommendation 28 October 2004.

[XPath20]
XML Path Language (XPath) 2.0. Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernández, Michael Kay, Jonathan Robie, and Jérôme Siméon, eds. W3C Recommendation 23 January 2007.

[XPathFunc]
XQuery 1.0 and XPath 2.0 Functions and Operators. Ashok Malhotra, Jim Melton, and Norman Walsh, eds. W3C Recommendation 23 January 2007.

[IRC Log July 21, 2008]
Joint meeting of OWL, RIF and I18N WGs..