W3C Working Group Note

# OWL 2 Web Ontology Language Manchester Syntax

## W3C Working Group Note 18 October 2012

**This version:**
http://www.w3.org/TR/2012/NOTE-owl2-manchester-syntax-20121018/
**Latest version:**
http://www.w3.org/TR/owl2-manchester-syntax/
**Previous version:**
http://www.w3.org/TR/2009/NOTE-owl2-manchester-syntax-20091027/
**Authors:**
Matthew Horridge, University of Manchester
Peter F. Patel-Schneider, Nuance Communications

A color-coded version of this document showing changes made since the previous version is also available.

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



---

## Abstract

The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents. The OWL 2 Document Overview describes the overall state of OWL 2, and should be read before other OWL 2 documents.

The Manchester syntax is a user-friendly compact syntax for OWL 2 ontologies; it is frame-based, as opposed to the axiom-based other syntaxes for OWL 2. The Manchester Syntax is used in the OWL 2 Primer, and this document provides the language used there. It is expected that tools will extend the Manchester Syntax for their own purposes, and tool builders may collaboratively extend the common language.

## 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#### May Be Superseded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/.*

#### Summary of Changes

There have been no substantive changes since the previous version. For details on the minor changes see the change log and color-coded diff.

#### Please Send Comments

Please send any comments to public-owl-comments@w3.org (public archive). Although work on this document by the OWL Working Group is complete, comments may be addressed in the errata or in future revisions. Open discussion among developers is welcome at public-owl-dev@w3.org (public archive).

#### No Endorsement

*Publication as a Working Group Note 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. The group does not expect this document to become a W3C Recommendation. 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.*

---

## Table of Contents

## 1 Introduction

The Manchester OWL syntax is a user-friendly syntax for OWL 2 descriptions, but it can also be used to write entire OWL 2 ontologies. The original version of the Manchester OWL syntax [*Manchester OWL DL Syntax*] was created for OWL 1 [*OWL Semantics and Abstract Syntax*]; it is here updated for OWL 2 [*OWL 2 Specification*]. The Manchester syntax is used in Protégé 4 [*Protégé 4*] and TopBraid Composer® [*TopBraid Composer*], particularly for entering and displaying descriptions associated with classes. Some tools (e.g., Protégé 4) extend the syntax to allow even more compact presentation in some situations (e.g., for explanation) or to replace IRIs by label values, but this document does not include any of these special-purpose extensions.

The Manchester OWL syntax gathers together information about names in a frame-like manner, as opposed to RDF/XML [*RDF Syntax*], the functional-style syntax for OWL 2 [*OWL 2 Specification*], and the XML syntax for OWL 2 [*OWL 2 XML Serialization*]. It is thus closer to the abstract syntax for OWL 1 [*OWL Semantics and Abstract Syntax*], than the above syntaxes for OWL 2. Nevertheless, parsing the Manchester OWL syntax into the OWL 2 structural specification is quite easy, as it is easy to identify the axioms inside each frame.

As the Manchester syntax is frame-based, it cannot directly handle all OWL 2 ontologies. However, there is a simple transform that will take any OWL 2 ontology that does not overload between object, data, and annotation properties or between classes and datatypes into a form that can be written in the Manchester syntax.

An example ontology in the Manchester OWL syntax can be found in the OWL Primer [*OWL 2 Primer*].

## 2 The Grammar

The Manchester syntax for OWL 2 ontologies is defined using a standard BNF notation, which is summarized in the table below.

**Table 1.** The BNF Notation Used in this Document

| Construct | Syntax | Example |
|---|---|---|
| non-terminal symbols | boldface | **ClassExpression** |
| terminal symbols | single quoted | `'PropertyRange'` |
| zero or more | curly braces | { **ClassExpression** } |
| zero or one | square brackets | [ **ClassExpression** ] |
| alternative | vertical bar | **Assertion** \| **Declaration** |
| grouping | parentheses | **dataPropertyExpression** ) |

Because comma-separated lists occur in very many places in the syntax, to save space the grammar has three meta-productions, one for non-empty lists, one for lists of minimum length two, and one for non-empty lists with annotations in them.

```
<NT>List ::= <NT> { ',' <NT> }
<NT>2List ::= <NT> ',' <NT>List
<NT>AnnotatedList ::= [annotations] <NT> { ',' [annotations] <NT> }
```

Documents in the Manchester OWL syntax form OWL 2 ontologies and consist of sequences of Unicode characters [*UNICODE*] and are encoded in UTF-8 [*RFC 3629*].

The grammar for the Manchester syntax does not explicitly show white space. White space is allowed between any two terminals or non-terminals except inside **nonNegativeInteger**, **prefixName**, **IRI**, and **literal**. White space is required between two terminals or non-terminals if its removal could cause ambiguity. Generally this means requiring white space except before and after punctuation (e.g., commas, parentheses, braces, and brackets).

White space is a sequence of blanks (U+20), tabs (U+9), line feeds (U+A), carriage returns (U+D), and comments. Comments are maximal sequences of Unicode characters starting with a '#' and not containing a line feed or a carriage return. Note that comments are only recognized where white space is allowed, and thus not inside the above non-terminals.

### 2.1 IRIs, Integers, Literals, and Entities

Names are IRIs (the successors of URIs) and can either be given in full or can be abbreviated similar to as in SPARQL [*SPARQL*]. Abbreviated IRIs consist of an optional colon-terminated prefix followed by a local part. Prefixes in abbreviated IRIs *must* not match any of the keywords of this syntax. Prefixes *should* begin with lower case letters so that they do not clash with colon-terminated keywords introduced in future versions of this syntax. Local parts with no prefix are expanded as if they had an initial colon and *must* not match any keyword of this syntax.

This syntax uses short forms for common data values, e.g., strings and numbers, and short forms for some common datatypes, e.g., integer. These correspond to the obvious long forms.

```
fullIRI := an IRI as defined in [RFC 3987], enclosed in a pair of < (U+3C) and > (U+3E) characters
prefixName := a finite sequence of characters matching the PNAME_NS production of [SPARQL] and not matching any of the keyword
terminals of the syntax
abbreviatedIRI := a finite sequence of characters matching the PNAME_LN production of [SPARQL]
simpleIRI := a finite sequence of characters matching the PN_LOCAL production of [SPARQL] and not matching any of the keyword
terminals of the syntax
IRI := fullIRI | abbreviatedIRI | simpleIRI

nonNegativeInteger ::= zero | positiveInteger
positiveInteger ::= nonZero { digit }
digits ::= digit { digit }
digit ::= zero | nonZero
nonZero := '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
zero ::= '0'
```

```
classIRI ::= IRI
Datatype ::= datatypeIRI | 'integer' | 'decimal' | 'float' | 'string'
datatypeIRI ::= IRI
objectPropertyIRI ::= IRI
dataPropertyIRI ::= IRI
annotationPropertyIRI ::= IRI
individual ::= individualIRI | nodeID
individualIRI ::= IRI
nodeID := a finite sequence of characters matching the BLANK_NODE_LABEL production of [SPARQL]

literal ::= typedLiteral | stringLiteralNoLanguage | stringLiteralWithLanguage | integerLiteral | decimalLiteral | floatingPointLiteral
typedLiteral ::= lexicalValue '^^' Datatype
stringLiteralNoLanguage ::= quotedString
stringLiteralWithLanguage ::= quotedString languageTag
languageTag := @ (U+40) followed a nonempty sequence of characters matching the langtag production from [BCP 47]
lexicalValue ::= quotedString
quotedString := a finite sequence of characters in which " (U+22) and \ (U+5C) occur only in pairs of the form \" (U+5C, U+22)
and \\ (U+5C, U+5C), enclosed in a pair of " (U+22) characters
floatingPointLiteral ::= [ '+' | '-'] ( digits ['.'digits] [exponent] | '.' digits[exponent]) ( 'f' | 'F' )
exponent ::= ('e' | 'E') ['+' | '-'] digits
decimalLiteral ::= ['+' | '-'] digits '.' digits
integerLiteral ::= ['+' | '-'] digits

entity ::= 'Datatype' '(' Datatype ')' | 'Class' '(' classIRI ')' | 'ObjectProperty' '(' objectPropertyIRI ')' | 'DataProperty'
'('dataPropertyIRI ')' | 'AnnotationProperty' '(' annotationPropertyIRI ')' | 'NamedIndividual' '(' individualIRI ')'
```

## 2.2 Ontologies and Annotations

```
annotations ::= 'Annotations:' annotationAnnotatedList
annotation ::= annotationPropertyIRI annotationTarget
annotationTarget ::= nodeID | IRI | literal

ontologyDocument ::=  { prefixDeclaration } ontology
prefixDeclaration ::= 'Prefix:' prefixName fullIRI
ontology ::= 'Ontology:' [ ontologyIRI [ versionIRI ] ] { import } { annotations } { frame }
ontologyIRI ::= IRI
versionIRI ::= IRI
import ::= 'Import:' IRI
frame ::= datatypeFrame | classFrame | objectPropertyFrame | dataPropertyFrame | annotationPropertyFrame | individualFrame | misc
```

The 'rdf:', 'rdfs:', 'owl:', and 'xsd:' prefixes are pre-defined as follows and cannot be changed. Each other prefix used in an ontology document must have exactly one prefix declaration in the ontology document.

```
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: owl: <http://www.w3.org/2002/07/owl#>
```

## 2.3 Property and Datatype Expressions

```
objectPropertyExpression ::= objectPropertyIRI | inverseObjectProperty
inverseObjectProperty ::= 'inverse' objectPropertyIRI
dataPropertyExpression ::= dataPropertyIRI

dataRange ::= dataConjunction 'or' dataConjunction { 'or' dataConjunction }
         | dataConjunction
dataConjunction ::= dataPrimary 'and' dataPrimary { 'and' dataPrimary }
         | dataPrimary
dataPrimary ::= [ 'not' ] dataAtomic
dataAtomic ::= Datatype
         | '{' literalList '}'
         | datatypeRestriction | '(' dataRange ')'
datatypeRestriction ::= Datatype '[' facet restrictionValue { ',' facet restrictionValue } ']'
facet ::= 'length' | 'minLength' | 'maxLength' | 'pattern' | 'langRange' | '<=' | '<' | '>=' | '>'
restrictionValue ::= literal
```

In a **datatypeRestriction**, the **facet**s and **restrictionValue**s must be valid for the **Datatype**, as in the OWL 2 Specification [*OWL 2 Specification*], after making the obvious change for the comparison facets.

## 2.4 Descriptions

The infix notation for descriptions is ambiguous as stated. This ambiguity is resolved in the usual way, with later productions in the description grammar below binding more tightly, so, for example,

```
p some a and p only b
```

is parsed as

```
(p some a) and (p only b)
```

```
description ::= conjunction 'or' conjunction { 'or' conjunction }
          | conjunction
conjunction ::= classIRI 'that' [ 'not' ] restriction { 'and' [ 'not' ] restriction }
          | primary 'and' primary { 'and' primary }
          | primary
primary ::= [ 'not' ] ( restriction | atomic )
restriction ::= objectPropertyExpression 'some' primary
          | objectPropertyExpression 'only' primary
          | objectPropertyExpression 'value' individual
          | objectPropertyExpression 'Self'
          | objectPropertyExpression 'min' nonNegativeInteger [ primary ]
          | objectPropertyExpression 'max' nonNegativeInteger [ primary ]
          | objectPropertyExpression 'exactly' nonNegativeInteger [ primary ]
          | dataPropertyExpression 'some' dataPrimary
          | dataPropertyExpression 'only' dataPrimary
          | dataPropertyExpression 'value' literal
          | dataPropertyExpression 'min' nonNegativeInteger [ dataPrimary ]
          | dataPropertyExpression 'max' nonNegativeInteger [ dataPrimary ]
          | dataPropertyExpression 'exactly' nonNegativeInteger [ dataPrimary ]
atomic ::= classIRI
          | '{' individualList '}'
          | '(' description ')'
```

### 2.5 Frames and Miscellaneous

```
datatypeFrame ::= 'Datatype:' Datatype
       { 'Annotations:'     annotationAnnotatedList }
       [ 'EquivalentTo:' annotations dataRange ]
       { 'Annotations:'     annotationAnnotatedList }

classFrame ::= 'Class:' classIRI
       { 'Annotations:'     annotationAnnotatedList
       | 'SubClassOf:'      descriptionAnnotatedList
       | 'EquivalentTo:'    descriptionAnnotatedList
       | 'DisjointWith:'    descriptionAnnotatedList
       | 'DisjointUnionOf:' annotations description2List }
       | 'HasKey:' annotations ( objectPropertyExpression | dataPropertyExpression )
                                         { objectPropertyExpression | dataPropertyExpression }


objectPropertyFrame ::= 'ObjectProperty:' objectPropertyIRI
       { 'Annotations:'     annotationAnnotatedList
       | 'Domain:'          descriptionAnnotatedList
       | 'Range:'           descriptionAnnotatedList
       | 'Characteristics:' objectPropertyCharacteristicAnnotatedList
       | 'SubPropertyOf:'   objectPropertyExpressionAnnotatedList
       | 'EquivalentTo:'    objectPropertyExpressionAnnotatedList
       | 'DisjointWith:'    objectPropertyExpressionAnnotatedList
       | 'InverseOf:'        objectPropertyExpressionAnnotatedList
       | 'SubPropertyChain:' annotations objectPropertyExpression 'o' objectPropertyExpression
                                                { 'o' objectPropertyExpression } }

objectPropertyCharacteristic ::= 'Functional' | 'InverseFunctional'
              | 'Reflexive' | 'Irreflexive' | 'Symmetric' | 'Asymmetric' | 'Transitive'

dataPropertyFrame ::= 'DataProperty:' dataPropertyIRI
       { 'Annotations:'     annotationAnnotatedList
       | 'Domain:'          descriptionAnnotatedList
       | 'Range:'           dataRangeAnnotatedList
       | 'Characteristics:' annotations 'Functional'
       | 'SubPropertyOf:'   dataPropertyExpressionAnnotatedList
       | 'EquivalentTo:'    dataPropertyExpressionAnnotatedList
       | 'DisjointWith:'    dataPropertyExpressionAnnotatedList }

annotationPropertyFrame ::= 'AnnotationProperty:' annotationPropertyIRI
       { 'Annotations:'     annotationAnnotatedList }
       | 'Domain:'          IRIAnnotatedList
       | 'Range:'           IRIAnnotatedList
       | 'SubPropertyOf:'   annotationPropertyIRIAnnotatedList

individualFrame ::= 'Individual:' individual
       { 'Annotations:'   annotationAnnotatedList
       | 'Types:'         descriptionAnnotatedList
       | 'Facts:'         factAnnotatedList
       | 'SameAs:'        individualAnnotatedList
       | 'DifferentFrom:' individualAnnotatedList }

fact ::= [ 'not' ] (objectPropertyFact | dataPropertyFact)
objectPropertyFact ::= objectPropertyIRI individual
dataPropertyFact ::= dataPropertyIRI literal

misc ::= 'EquivalentClasses:' annotations description2List
       | 'DisjointClasses:' annotations description2List
       | 'EquivalentProperties:' annotations objectProperty2List
       | 'DisjointProperties:' annotations objectProperty2List
       | 'EquivalentProperties:' annotations dataProperty2List
       | 'DisjointProperties:' annotations dataProperty2List
       | 'SameIndividual:' annotations individual2List
       | 'DifferentIndividuals:' annotations individual2List
```

W3C Working Group Note

## 2.6 Global Concerns

The Manchester syntax has the same global conditions on ontologies as for OWL 2 ontologies in the OWL 2 Specification [*OWL 2 Specification*], with the addition of the typing constraints for OWL 2 DL ontologies, but using the appropriate frame instead of declarations.

The Manchester syntax global conditions for OWL 2 DL ontologies are the same as in the OWL 2 Specification except as mentioned just above.

# 3 Quick Reference

This is a made-up partial ontology that provides a quick reference guide to the Manchester Syntax. Not all of the ontology makes logical sense so that all aspects of the syntax can be shown in a small example.

All colon-terminated keyword constructs except Ontology: (e.g., Import:, Class:, Domain:, SubClassOf:) are optional and can be repeated. Most keyword constructs take a comma-separated list of sub-constructs, which is sometimes indicated by ",...". Annotations are allowed for elements in these lists of sub-constructs except where annotations are explicitly noted (e.g., in DisjointUnionOf:, in DisjointClasses:).

```
Prefix: : <http://ex.com/owl/families#>
Prefix: g: <http://ex.com/owl2/families#>

Ontology: <http://example.com/owl/families> <http://example.com/owl/families-v1>
  Import: <http://ex.com/owl2/families.owl>
  Annotations: creator John,
               Annotations: rdfs:comment "Creation Year"
                 creationYear 2008,
               mainClass Person

  ObjectProperty: hasWife
    Annotations: ...
    Characteristics: Functional, InverseFunctional, Reflexive, Irreflexive, Asymmetric, Transitive
    Domain: Annotations: rdfs:comment "General domain",
                         creator John
              Person,
            Annotations: rdfs:comment "More specific domain"
              Man
    Range: Person, Woman
    SubPropertyOf: hasSpouse, loves
    EquivalentTo: isMarriedTo ,...
    DisjointWith: hates ,...
    InverseOf: hasSpouse, inverse hasSpouse
    SubPropertyChain: Annotations: ... hasChild o hasParent o...

  DataProperty: hasAge
    Annotations: ...
    Characteristics: Functional
    Domain: Person ,...
    Range: integer ,...
    SubPropertyOf: hasVerifiedAge ,...
    EquivalentTo: hasAgeInYears ,...
    DisjointWith: hasSSN ,...

  AnnotationProperty: creator
    Annotations: ...
    Domain: Person ,...
    Range: integer ,...
    SubPropertyOf: initialCreator ,...

  Datatype: NegInt
    Annotations: ...
    EquivalentTo: integer[< 0]

  Class: Person
    Annotations: ...
    SubClassOf: owl:Thing that hasFirstName exactly 1 and hasFirstName only string[minLength 1]  ,...
    SubClassOf: hasAge exactly 1 and hasAge only not NegInt,...
    SubClassOf: hasGender exactly 1 and hasGender only {female , male} ,...
    SubClassOf: hasSSN max 1, hasSSN min 1
    SubClassOf: not hates Self, ...
    EquivalentTo: g:People ,...
    DisjointWith: g:Rock , g:Mineral ,...
    DisjointUnionOf: Annotations: ... Child, Adult
    HasKey: Annotations: ... hasSSN

  Individual: John
    Annotations: ...
    Types: Person , hasFirstName value "John" or hasFirstName value "Jack"^^xsd:string
    Facts: hasWife Mary, not hasChild Susan, hasAge 33, hasChild _:child1
    SameAs: Jack ,...
    DifferentFrom: Susan ,...

  Individual: _:child1
    Annotations: ...
    Types: Person ,...
    Facts: hasChild Susan ,...

  DisjointClasses: Annotations: ... g:Rock, g:Scissor, g:Paper
  EquivalentProperties: Annotations: ... hates, loathes, despises
  DisjointProperties: Annotations: ... hates, loves, indifferent
  EquivalentProperties: Annotations: ... favoriteNumber, g:favouriteNumber, g:favouriteInteger
  DisjointProperties: Annotations: ... favoriteInteger, favouriteReal
  SameIndividual: Annotations: ... John, Jack, Joe, Jim
  DifferentIndividuals: Annotations: ... John, Susan, Mary, Jill
```

## 4 Appendix: Translation to and from OWL 2 Functional-Style Syntax

Most of the translation between the Manchester OWL syntax and OWL 2 is obvious. The translation given here is with the OWL 2 Functional-Style Syntax [*OWL 2 Specification*].

### 4.1 Informal Description

In many cases there is a one-to-one correspondence between the Manchester OWL syntax and the OWL 2 Functional-Style Syntax. For example, **dataComplementOf** in the Manchester OWL syntax corresponds directly to **dataComplementOf** in the OWL 2 Functional-Style Syntax. All that is required is to translate the keywords and adjust to a parenthesized syntax.

IRIs and their parts are the same in the Manchester OWL syntax and the OWL 2 Functional-Style Syntax, no change is needed for them, except that the "special" datatypes are translated into the corresponding XML Schema datatypes. Literals are mostly the same, but the abbreviated syntaxes for numbers and strings have to be translated in the obvious way. The syntax for data ranges in the Manchester OWL syntax corresponds exactly with the syntax in the OWL 2 Functional-Style Syntax.

The syntax for annotations in the Manchester OWL syntax closely corresponds to the syntax in the OWL 2 Functional-Style Syntax. The only special processing that needs to be done is to determine which frame to attach entity annotations to in the reverse mapping. Translating to the Functional-Style syntax and back again can thus lose some non-logical information in the Manchester syntax.

Descriptions also correspond closely between the Manchester OWL syntax and the OWL 2 Functional-Style Syntax.

The translation of frame axioms is performed by splitting them into pieces that correspond to single axioms. This is done by taking each of the pieces of the frame (Annotations:, Domain:, Range:, etc) and making new frames for each of them. The new frame is of the same kind (Class:, ObjectProperty:, etc.) and for the same IRI. Then each resultant frame that contains an AnnotatedList with more than one element is broken into a frame for each element of the list in a similar manner.

The resultant axioms and any miscellaneous axioms then correspond closely to the OWL 2 Functional-Style Syntax axioms and can be directly translated. The only special cases are that annotations directly in frames become annotations in entity annotation axioms and that (negative) property assertions have to be disambiguated depending on whether the property is an object property or a data property.

Translations of OWL 2 Functional-Style Syntax axioms back to frames can be done piecemeal or the axioms on a single entity can be all combined together, which is done here.

The remaining top-level constructs of an ontology (prefix declarations, imports, ontology annotations, and the ontology name) can be directly translated.

### 4.2 Formal Description for Mapping to OWL 2 Functional-Style Syntax

Formally the transformation takes an ontology in the Manchester OWL syntax and produces an ontology in the Functional-Style syntax. The transformation needs access to the imported ontologies.

First, for each frame in the ontology, produce the appropriate declaration as follows:

| Frame | Declaration |
|---|---|
| Class: *IRI* ... | Declaration( Class(*IRI*) ) |
| ObjectProperty: *IRI* ... | Declaration( ObjectProperty(*IRI*) ) |
| DataProperty: *IRI* ... | Declaration( DataProperty(*IRI*) ) |
| AnnotationProperty: *IRI* ... | Declaration( AnnotationProperty(*IRI*) ) |
| Individual: *IRI* ... | Declaration( NamedIndividual(*IRI*) ) |
| Individual: *nodeID* ... | |

Second, split up frames into single axioms in three stages. The first stage splits apart top-level pieces of frames that have multiple top-level pieces, transforming F: *IRI p1 p2* ... into F: *IRI p1* F: *IRI p2* ... for F: one of the frame keywords (Class:, ...), until no more transformations are possible. The second stage splits apart the pieces of each of the top-level pieces, transforming F: *IRI* P: *s1 s2* ... into F: *IRI* P: *s1* F: *IRI* P: *s2* ... for P: one of the keywords immediately inside a frame (Annotations:, SubClassOf:, ...), until no more transformations are possible. The third stage just removes any frame containing only an IRI.

Next, perform the actual syntax transformation. Any piece of syntax with no transformation listed here is just copied through.

| Nonterminal | Form | Transformation (*T*) |
|---|---|---|
| **simpleIRI** | *localPart* | :*localPart* |
| **Datatype** | integer | xsd:integer |
| **Datatype** | decimal | xsd:decimal |
| **Datatype** | float | xsd:float |
| **Datatype** | string | xsd:string |
| **integerLiteral** | *integer* | "*integer*"^^xsd:integer |
| **decimalLiteral** | *decimal* | "*decimal*"^^xsd:decimal |
| **floatingPointLiteral** | *float* | "*float*"^^xsd:float |
| **stringLiteralNoLanguage** | *string* | *string* |
| **stringLiteralWithLanguage** | *string@tag* | *string@tag* |
| **facet** | length | xsd:length |
| **facet** | minLength | xsd:minLength |
| **facet** | maxLength | xsd:maxLength |
| **facet** | pattern | xsd:pattern |
| **facet** | langRange | rdf:langRange |
| **facet** | <= | xsd:minInclusive |
| **facet** | < | xsd:minExclusive |
| **facet** | >= | xsd:maxInclusive |
| **facet** | > | xsd:maxExclusive |
| **datatypeRestriction** | *Datatype*[*facet-value list*] | DatatypeRestriction(*T(datatype) T(facet-value list)*) |
| **dataAtomic** | { *literal list* } | DataOneOf(*T(literal list)*) |
| **dataAtomic** | (*dataRange*) | *T(dataRange)* |
| **dataPrimary** | *dataAtomic* | *T(dataAtomic)* |
| **dataPrimary** | not *dataAtomic* | DataComplementOf(*T(dataAtomic)*) |
| **dataConjunction** | *dataPrimary* and ... | DataIntersectionOf(*T(dataPrimary)* ...) |

| | | |
|---|---|---|
| **dataConjunction** | *dataPrimary* | *T(dataPrimary)* |
| **dataRange** | *dataConjunction* or ... | DataUnionOf(*T(dataConjunction)* ...) |
| **dataRange** | *dataConjunction* | *T(dataConjunction)* |
| **inverseObjectProperty** | inverse *objectPropertyExpression* | InverseObjectProperty(*T(objectPropertyExpression)*) |
| **atomic** | {*individual list*} | ObjectOneOf(*T(individual list)*) |
| **atomic** | (*description*) | *T(description)* |
| **restriction** | *objectPropertyExpression* some *primary* | ObjectSomeValuesFrom(*T(objectPropertyExpression) T(primary)*) |
| **restriction** | *objectPropertyExpression* only *primary* | ObjectAllValuesFrom(*T(objectPropertyExpression) T(primary)*) |
| **restriction** | *objectPropertyExpression* value *individual* | ObjectHasValue(*T(objectPropertyExpression) individual*) |
| **restriction** | *objectPropertyExpression* min *nni* | ObjectMinCardinality(*T(objectPropertyExpression) nni*) |
| **restriction** | *objectPropertyExpression* min *nni primary* | ObjectMinCardinality(*T(objectPropertyExpression) nni T(primary)*) |
| **restriction** | *objectPropertyExpression* exactly *nni* | ObjectExactCardinality(*T(objectPropertyExpression) nni*) |
| **restriction** | *objectPropertyExpression* exactly *nni primary* | ObjectExactCardinality(*T(objectPropertyExpression) nni T(primary)*) |
| **restriction** | *objectPropertyExpression* max *nni* | ObjectMaxCardinality(*T(objectPropertyExpression) nni*) |
| **restriction** | *objectPropertyExpression* max *nni primary* | ObjectMaxCardinality(*T(objectPropertyExpression) nni T(primary)*) |
| **restriction** | *objectPropertyExpression* Self | ObjectHasSelf(*T(objectPropertyExpression)*) |
| **restriction** | *dataPropertyExpression* some *dataRange* | DataSomeValuesFrom(*T(dataPropertyExpression) T(dataRange)*) |
| **restriction** | *dataPropertyExpression* only *dataRange* | DataAllValuesFrom(*T(dataPropertyExpression) T(dataRange)*) |
| **restriction** | *dataPropertyExpression* value *literal* | DataHasValue(*T(dataPropertyExpression) T(literal)*) |
| **restriction** | *dataPropertyExpression* min *nni* | DataMinCardinality(*T(dataPropertyExpression) nni*) |
| **restriction** | *dataPropertyExpression* min *nni dataRange* | DataMinCardinality(*T(dataPropertyExpression) nni T(dataRange)*) |
| **restriction** | *dataPropertyExpression* exactly *nni* | DataExactCardinality(*T(dataPropertyExpression) nni*) |
| **restriction** | *dataPropertyExpression* exactly *nni dataRange* | DataExactCardinality(*T(dataPropertyExpression) nni T(dataRange)*) |
| **restriction** | *dataPropertyExpression* max *nni* | DataMaxCardinality(*T(dataPropertyExpression) nni*) |
| **restriction** | *dataPropertyExpression* max *nni dataRange* | DataMaxCardinality(*T(dataPropertyExpression) nni T(dataRange)*) |
| **primary** | *atomic* | *T(atomic)* |
| **primary** | not *atomic* | ObjectComplementOf(*T(atomic)*) |
| **conjunction** | *classIRI* that *primary* ... | ObjectIntersectionOf(*classIRI T(primary)* ...) |
| **conjunction** | *primary* and ... | ObjectIntersectionOf(*T(primary)* ...) |
| **conjunction** | *primary* | *T(primary)* |
| **description** | *conjunction* or ... | ObjectUnionOf(*T(conjunction)* ...) |
| **description** | *conjunction* | *T(conjunction)* |
| **annotation** | *annotations annotationPropertyIRI target* | Annotation(T(*annotations*) *annotationPropertyIRI* T(*target*)) |
| **annotations** | | |
| **annotations** | Annotations: *annotation* ... | Annotation(T(*annotation*) ... |
| **datatypeFrame** | Datatype: *Datatype* Annotations: *annotations annotationPropertyIRI target* | AnnotationAssertion(T(*annotations*) *annotationPropertyIRI* T(*Datatype*) T(*target*)) |
| **datatypeFrame** | Datatype: *IRI* EquivalentTo: *annotations dataRange* | DatatypeDefinition(*T(annotations) IRI T(dataRange)*) |
| **classFrame** | Class: *IRI* Annotations: *annotations annotationPropertyIRI target* | AnnotationAssertion(T(*annotations*) *annotationPropertyIRI IRI* T(*target*)) |
| **classFrame** | Class: *IRI* SubClassOf: *annotations description* | SubClassOf(*T(annotations) IRI T(description)*) |
| **classFrame** | Class: *IRI* EquivalentTo: *annotations description* | EquivalentClasses(*T(annotations) IRI T(description)*) |
| **classFrame** | Class: *IRI* DisjointWith: *annotations description* | DisjointClasses(*T(annotations) IRI T(description)*) |
| **classFrame** | Class: *IRI* DisjointUnionOf: *annotations descriptions* | DisjointUnion(*T(annotations) IRI T(description)*) |
| **classFrame** | Class: *IRI* HasKey: *annotations properties* | HasKey(*T(annotations) IRI T(properties)*) |
| **other** | *properties* | ( *T(objectProperties)* ) ( *T(dataProperties)* ) Note: Sort the properties into object and data properties. |
| **objectPropertyFrame** | ObjectProperty: *IRI* Annotations: *annotations annotationPropertyIRI target* | AnnotationAssertion(T(*annotations*) *annotationPropertyIRI IRI* T(*target*)) |
| **objectPropertyFrame** | ObjectProperty: *IRI* Domain: *annotations description* | ObjectPropertyDomain(*T(annotations) IRI T(description)*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* Range: *annotations description* | ObjectPropertyRange(*T(annotations) IRI T(description)*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* Characteristics: *annotations* Functional | ObjectFunctionalProperty(*T(annotations) IRI*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* Characteristics: *annotations* InverseFunctional | ObjectInverseFunctionalProperty(*T(annotations) IRI*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* Characteristics: *annotations* Reflexive | ObjectReflexiveProperty(*T(annotations) IRI*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* Characteristics: *annotations* Irreflexive | ObjectIrreflexiveProperty(*T(annotations) IRI*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* Characteristics: *annotations* Symmetric | ObjectSymmetricProperty(*T(annotations) IRI*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* Characteristics: *annotations* Asymmetric | ObjectAsymmetricProperty(*T(annotations) IRI*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* Characteristics: *annotations* Transitive | ObjectTransitiveProperty(*T(annotations) IRI*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* SubPropertyOf: *annotations* objectPropertyExpression | SubObjectPropertyOf(*T(annotations) IRI T(objectPropertyExpression)*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* EquivalentTo: *annotations* objectPropertyExpression | EquivalentObjectProperties(*T(annotations) IRI T(objectPropertyExpression)*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* DisjointWith: *annotations* objectPropertyExpression | DisjointObjectProperties(*T(annotations) IRI T(objectPropertyExpression)*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* InverseOf: *annotations* objectPropertyExpression | InverseObjectProperties(*T(annotations) IRI T(objectPropertyExpression)*) |
| **objectPropertyFrame** | ObjectProperty: *IRI* SubPropertyChain: *objectPropertyExpression* o ... | SubObjectPropertyOf(ObjectPropertyChain(*T(objectPropertyExpression)* ...) *IRI*) |
| **dataPropertyFrame** | DataProperty: *IRI* Annotations: *annotations annotationPropertyIRI target* | AnnotationAssertion(T(*annotations*) *annotationPropertyIRI IRI* T(*target*)) |
| **dataPropertyFrame** | DataProperty: *IRI* Domain: *annotations description* | DataPropertyDomain(*T(annotations) IRI T(description)*) |
| **dataPropertyFrame** | DataProperty: *IRI* Range: *annotations dataRange* | DataPropertyRange(*T(annotations) IRI T(dataRange)*) |
| **dataPropertyFrame** | DataProperty: *IRI* Characteristics: *annotations* Functional | FunctionalDataProperty(*T(annotations) IRI*) |
| **dataPropertyFrame** | DataProperty: *IRI* SubPropertyOf: *annotations* dataPropertyExpression | SubDataPropertyOf(*T(annotations) IRI T(dataPropertyExpression)*) |
| **dataPropertyFrame** | DataProperty: *IRI* EquivalentTo: *annotations* dataPropertyExpression | EquivalentDataProperties(*T(annotations) IRI T(dataPropertyExpression)*) |
| **dataPropertyFrame** | DataProperty: *IRI* DisjointWith: *annotations* dataPropertyExpression | DisjointDataProperties(*T(annotations) IRI T(dataPropertyExpression)*) |
| **annotationPropertyFrame** | AnnotationProperty: *IRI* Annotations: *annotations annotationPropertyIRI target* | AnnotationAssertion(T(*annotations*) *annotationPropertyIRI IRI* T(*target*)) |
| **annotationPropertyFrame** | AnnotationProperty: *IRI* Domain: *annotations IRI* | AnnotationPropertyDomain(*T(annotations) IRI IRI*) |
| **annotationPropertyFrame** | AnnotationProperty: *IRI* Range: *annotations IRI* | AnnotationPropertyRange(*T(annotations) IRI IRI*) |
| **annotationPropertyFrame** | AnnotationProperty: *IRI* SubPropertyOf: *annotations* annotationPropertyIRI | SubAnnotationPropertyOf(*T(annotations) IRI T(annotationPropertyIRI)*) |
| **individualFrame** | Individual: *IRI* Annotations: *annotations annotationPropertyIRI target* | AnnotationAssertion(T(*annotations*) *annotationPropertyIRI IRI* T(*target*)) |
| **individualFrame** | Individual: *nodeID* Annotations: *annotations annotation* | AnnotationAssertion(T(*annotations*) *annotationPropertyIRI nodeID* T(*target*)) |
| **individualFrame** | Individual: *individual* Types: *annotations description* | ClassAssertion(*T(annotations) T(description) individual*) |
| **individualFrame** | Individual: *individual* Facts: *annotations objectPropertyIRI individual2* | ObjectPropertyAssertion(*T(annotations) objectPropertyIRI individual individual2*) |
| **individualFrame** | Individual: *individual* Facts: *annotations* not *objectPropertyIRI individual2* | NegativeObjectPropertyAssertion(*T(annotations) objectPropertyIRI individual individual2*) |

| | | |
|---|---|---|
| **individualFrame** | Individual: *individual* Facts: *annotations dataPropertyIRI literal* | DataPropertyAssertion(*T(annotations)* dataPropertyIRI individual T(literal)) |
| **individualFrame** | Individual: *individual* Facts: *annotations* not *dataPropertyIRI literal* | NegativeDataPropertyAssertion(*T(annotations)* dataPropertyIRI individual T(literal)) |
| **individualFrame** | Individual: *individual* SameAs: *annotations individual2* | SameIndividual(*T(annotations) individual individual2*) |
| **individualFrame** | Individual: *individual* DifferentFrom: *annotations individual2* | DifferentIndividuals(*T(annotations) individual individual2*) |
| **misc** | EquivalentClasses: *annotations descriptions* | EquivalentClasses(*T(annotations) T(descriptions)*) |
| **misc** | DisjointClasses: *annotations descriptions* | DisjointClasses(*T(annotations) T(descriptions)*) |
| **misc** | EquivalentProperties: *annotations objectProperties* | EquivalentObjectProperties(*T(annotations) T(objectProperties)*) |
| **misc** | DisjointProperties: *annotations objectProperties* | DisjointObjectProperties(*T(annotations) T(objectProperties)*) |
| **misc** | EquivalentProperties: *annotations dataProperties* | EquivalentDataProperties(*T(annotations) T(dataProperties)*) |
| **misc** | DisjointProperties: *annotations dataProperties* | DisjointDataProperties(*T(annotations) T(dataProperties)*) |
| **misc** | SameIndividual: *annotations individuals* | SameIndividual(*T(annotations) individuals*) |
| **misc** | DifferentIndividuals: *annotations individuals* | DifferentIndividuals(*T(annotations) individuals*) |
| **prefixDeclaration** | Prefix: *prefix fullIRI* | Prefix(*prefix = fullIRI*) |
| **import** | Import: *IRI* | Import(*IRI*) |
| **ontology** | Ontology: *IRI IRI imports annotations frames* | Ontology(*IRI IRI T(imports) T(annotations) T(frames)*) |
| **ontology** | Ontology: *IRI imports annotations frames* | Ontology(*IRI T(imports) T(annotations) T(frames)*) |
| **ontology** | Ontology: *imports annotations frames* | Ontology(*T(imports) T(annotations) T(frames)*) |
| **ontologyDocument** | *prefixDeclarations ontology* | *T(prefixDeclarations) T(ontology)* |

Finally, put the declarations produced in the first step into the ontology.

### 4.3 Formal Description for Mapping from OWL 2 Functional-Style Syntax

The mapping from the Functional-Style Syntax back to the Manchester Syntax essentially just runs the above translation in reverse.

Some axioms that become part of a frame in the Manchester syntax do not need to have a name for the frame, e.g., a SubClassOf axiom between two complex descriptions, so the construction below cannot be directly used. To transform these axioms to the Manchester syntax, take a fresh name and turn the axiom into two axioms, one that makes the new name equivalent to the first piece of the axiom and the other the axiom with the sub-construct replaced by the new name. This would turn a SubClassOf axiom into an EquivalentClasses axiom plus a SubClassOf axiom.

The basic mapping first creates a trivial frame containing only an IRI for each named class, property, and individual in the ontology. Second, turn the Functional-Style Syntax into the Manchester Syntax by running the syntax transformation above in reverse. The non-determinism in the mapping of entity annotations is resolved by uniformly making them annotations in individual frames. Third, collapse frames for the same entity into one frame by running that part of the forward transformation in reverse. This step does not affect the meaning of an ontology and is thus optional.

## 5 Appendix: Internet Media Type, File Extension and Macintosh File Type

**Contact**
: Ivan Herman / Sandro Hawke

**See also**
: How to Register a Media Type for a W3C Specification [*Register MIME*] and Internet Media Type registration, consistency of use [*MIME Consistency*].

The Internet Media Type / MIME Type for the OWL Manchester Syntax is "text/owl-manchester".

It is recommended that OWL Manchester Syntax files have the extension ".omn" (all lowercase) on all platforms.

It is recommended that OWL Manchester Syntax files stored on Macintosh HFS file systems be given a file type of "TEXT".

The information that follows will be submitted to the IESG for review, approval, and registration with IANA.

**Type name**
: text

**Subtype name**
: owl-manchester

**Required parameters**
: None

**Optional parameters**
: charset This parameter may be required when transfering non-ascii data across some protocols. If present, the value of charset is always UTF-8.

**Encoding considerations**
: The syntax of the OWL Manchester Syntax is expressed over code points in Unicode [*UNICODE*]. The encoding is always UTF-8 [*RFC 3629*].

**Security considerations**
: The OWL Manchester Syntax uses IRIs as term identifiers. Applications interpreting data expressed in the OWL Manchester Syntax should address the security issues of Internationalized Resource Identifiers (IRIs) [*RFC 3987*] Section 8, as well as Uniform Resource Identifiers (URI): Generic Syntax [*RFC3986*] Section 7. Multiple IRIs may have the same appearance. Characters in different scripts may look similar (a Cyrillic "o" may appear similar to a Latin "o"). A character followed by combining characters may have the same visual representation as another character (LATIN SMALL LETTER E followed by COMBINING ACUTE ACCENT has the same visual representation as LATIN SMALL LETTER E WITH ACUTE). Any person or application that is writing or interpreting data in the OWL Manchester Syntax must take care to use the IRI that matches the intended semantics, and avoid IRIs that may look similar. Further information about matching of similar characters can be found in Unicode Security Considerations [*UNISEC*] and Internationalized Resource Identifiers (IRIs) [*RFC 3987*] Section 8.

**Interoperability considerations**
: There are no known interoperability issues.

**Published specification**
: This specification.

**Applications which use this media type**
: This media type is used by Protege 4.

**Additional information**
: None.

**Magic number(s)**
: OWL Manchester Syntax documents may have the strings 'Prefix:' or 'Ontology:' (case dependent) near the beginning of the document.

**File extension(s)**
: ".omn"

**Base URI**
: There are no constructs in the OWL Manchester Syntax to change the Base URI.

**Macintosh file type code(s)**
: "TEXT"

**Person & email address to contact for further information**
: Ivan Herman, ivan@w3.org / Sandro Hawke, sandro@w3.org. Please send technical comments and questions about OWL to public-owl-comments@w3.org, a mailing list with a public archive at http://lists.w3.org/Archives/Public/public-owl-comments/

**Intended usage**

W3C Working Group Note

COMMON

**Restrictions on usage**
None

**Author/Change controller**
The OWL Mancheser Syntax is the product of the W3C OWL Working Group in cooperation with OWL ontology tool builders; the specification may be extended by groups of OWL tool builders; W3C reserves change control over this specification.

## 6 Appendix: Change Log (Informative)

### 6.1 Changes Since Working Group Note

This section summarizes the changes to this document since the Working Group Note of 27 October, 2009.

- With the publication of the XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes Recommendation of 5 April 2012, the elements of OWL 2 which are based on XSD 1.1 are now considered required, and the note detailing the optional dependency on the XSD 1.1 Candidate Recommendation of 30 April, 2009 has been removed from the "Status of this Document" section.
- A bug in the syntax was fixed by replacing rdf:langPattern with rdf:langRange throughout the document (see OWL 2 Errata page).

### 6.2 Changes Since Previous Working Draft

This section summarizes the changes to this document since the Working Draft of 11 June, 2009.

- The names of two non-terminals were changed. This change does not affect the language and was made to align the names of the non-terminals with the names used elsewhere.

## 7 References

### 7.1 Normative References

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

**[OWL 2 Specification]**
*OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax* Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, eds. W3C Proposed Edited Recommendation, 18 October 2012, http://www.w3.org/TR/2012/PER-owl2-syntax-20121018/. Latest version available at http://www.w3.org/TR/owl2-syntax/.

**[RDF Test Cases]**
*RDF Test Cases*. Jan Grant and Dave Beckett, eds. W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/. Latest version available as http://www.w3.org/TR/rdf-testcases/.

**[RFC 3629]**
*RFC 3629: UTF-8, a transformation format of ISO 10646*. F. Yergeau. IETF, November 2003, http://www.ietf.org/rfc/rfc3629.txt

**[RFC 3987]**
*RFC 3987: Internationalized Resource Identifiers (IRIs)*. M. Duerst and M. Suignard. IETF, January 2005, http://www.ietf.org/rfc/rfc3987.txt

**[SPARQL]**
*SPARQL Query Language for RDF*. Eric Prud'hommeaux and Andy Seaborne, eds. W3C Recommendation, 15 January 2008, http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/. Latest version available as http://www.w3.org/TR/rdf-sparql-query/.

**[UNICODE]**
*The Unicode Standard*. The Unicode Consortium, Version 5.1.0, ISBN 0-321-48091-0, as updated from time to time by the publication of new versions. (See http://www.unicode.org/unicode/standard/versions/ for the latest version and additional information on versions of the standard and of the Unicode Character Database).

### 7.2 Non-normative References

**[Manchester OWL DL Syntax]**
*The Manchester OWL Syntax*. Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens, and Hai H. Wang. OWL Experiences and Directions Workshop, 2006.

**[MIME Consistency]**
*Internet Media Type registration, consistency of use*. Tim Bray, ed. W3C TAG Finding, 30 April 2004.

**[OWL 1 Semantics and Abstract Syntax]**
*OWL Web Ontology Language: Semantics and Abstract Syntax*. Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks, eds. W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-semantics-20040210/. Latest version available at http://www.w3.org/TR/owl-semantics/.

**[OWL 2 Primer]**
*OWL 2 Web Ontology Language: Primer* Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, Sebastian Rudolph, eds. W3C Proposed Edited Recommendation, 18 October 2012, http://www.w3.org/TR/2012/PER-owl2-primer-20121018/. Latest version available at http://www.w3.org/TR/owl2-primer/.

**[OWL 2 XML Serialization]**
*OWL 2 Web Ontology Language: XML Serialization* Boris Motik, Bijan Parsia, Peter F. Patel-Schneider, eds. W3C Proposed Edited Recommendation, 18 October 2012, http://www.w3.org/TR/2012/PER-owl2-xml-serialization-20121018/. Latest version available at http://www.w3.org/TR/owl2-xml-serialization/.

**[Protégé 4]**
*Protégé 4 User Documentation*. October 2008. http://protegewiki.stanford.edu/index.php/Protege4UserDocs

**[RDF Syntax]**
*RDF/XML Syntax Specification (Revised)*. Dave Beckett, ed. W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/. Latest version available as http://www.w3.org/TR/rdf-syntax-grammar/.

**[Register MIME]**
*Register an Internet Media Type for a W3C Spec*. Philippe Le Hégaret, ed. W3C Guidebook.

**[RFC 3986]**
*RFC 3986: Uniform Resource Identifier (URI): Generic Syntax*. T. Berners-Lee, R. Fielding, and L. Masinter. IETF, January 2005, http://www.ietf.org/rfc/rfc3986.txt

**[TopBraid Composer]**
*TopBraid Composer Home Page*. TopQuadrant, October 2008. http://www.topquadrant.com/topbraid/composer/

**[UNISEC]**
*Unicode Security Considerations*. Mark Davis and Michel Suignard. Unicode technical report 36, 23 July 2008, http://www.unicode.org/reports/tr36/tr36-7.html. Latest version available as http://www.unicode.org/reports/tr36/.