

# RDF Facets

## Bringing the Semantic Web to Traditional Web Search Engines

November 25, 2004

Patrick Stickler  
patrick.stickler@nokia.com

### Overview

RDF uses URIs. URIs tend to cause problems with most token keyword based search engines due to special characters which are allowed in URIs but typically ignored/discarded/disallowed in keywords by most web search engines. And unlike simple token keywords, RDF knowledge is expressed as structured property+value pairs, therefore there must be some means of expressing property+value pairs as single, unified string keywords escaped in a manner acceptable to a broad range of web search engines.

Furthermore, RDF provides for arbitrary datatypes, and so reliable/meaningful comparisons of values can be difficult in practice, since while we want a fairly flexible and future proof metadata framework for expressing knowledge, most search engines support only a very basic set of actual datatypes.

A solution that is flexible and scalable, supporting the evolution, extension, and adoption of various vocabularies, must be able to deal with arbitrary URIs, property+value pairs, datatyped values, and language qualifications without major retooling of the underlying infrastructure every time vocabularies and ontologies change. Ideally, a system would be automatically configured and updated based on knowledge defined in RDF itself.

The following is a description of "RDF Facets", an intermediate, simplified, and normalized representation of (a subset of) RDF modeled knowledge which should be suitable for efficient use with most token keyword based web search engines (as well as more capable engines supporting a broader range of basic datatypes such as numbers and dates).

### A Little Bit About RDF

It's not necessary to go into all the gory details of RDF to understand how to encode and use RDF Facets. The essential bit to understand is that RDF is all about statements describing things.

An RDF statement has three parts:

1. The thing being described (the "subject").
2. The characteristic/property being attributed to that thing (the "predicate").
3. The realization/specification/value of that property (the "object").

So an RDF statement is simply

subject + predicate + object

RDF statements are also called "triples", due to their three-part structure.

Subjects and predicates are (for our purposes) denoted by URIs. Objects can be URIs, simple strings, language qualified strings, XML content, or datatyped values.

E.g. if we have a ringtone identified by the URI `http://example.org/coolRingtone` and the ringtone had a title "A very cool ringtone" defined using the Dublin Core (DC) vocabulary title property, then the RDF statement (triple) would look like

```
<http://example.org/coolRingtone> <http://purl.org/dc/elements/1.1/title> "A very cool ringtone" .
```

Because datatypes are essential for reliable and meaningful comparison of values, we need to provide for a select few core, essential value types; namely strings, numbers, dates and times, XML fragments, and of course URIs. It's not practical to try to support any arbitrary datatypes, so any datatyped values which cannot be coerced into one of these core types are simply treated as "literal" raw values, where only equality can be tested.

Ideally, the mappings from various datatypes to the RDF Facet core types would be defined in RDF, such that an agent which converts RDF graphs to RDF Facets would utilize that knowledge in determining which mappings were appropriate for each particular property value. This mapping information could also, of course, be hard coded in the agent, but would of course require modifying the code to support additional datatypes.

(For those familiar with RDF, note that blank nodes are not (yet) supported in RDF Facets, and hence there is no representation of system identifiers)

In addition to the basic statement (triple) structure of RDF, it is also possible to qualify statements via additional statements about statements. This is called reification. In FN, we use reification to assign a relevance percentage to topic classifications (e.g. a given resource may be 30% relevant to Java and 80% relevant to MMS).

A reification is a set of statements, which specify the subject, predicate, and object of another statement, along with additional statements qualifying the reified statement (see example below).

RDF Facets are a means to encode the predicate+object portion of an RDF statement as an alphanumeric string without loss of the boundary between the property and its value and with a fundamental degree of datatype support.

A description of a URI denoted resource, including statement reifications (but excluding statements with blank node objects) can then be distilled down into a set of normalized facets based on the core set of value types, which are all that any given search engine need support.

## RDF Index Facet Syntax

The syntax for RDF Index Facets is:

```
DESCRIPTION      = SUBJECT ( WHITESPACE+ INDEX_FACET )+
INDEX_FACET      = PROPERTY_VALUE REIFICATION*
PROPERTY_VALUE   = PREDICATE OBJECT
SUBJECT          = "<" URI ">"
PREDICATE        = "x" FACET_ENCODED_URI "X"
OBJECT           = ( U_OBJECT | N_OBJECT | S_OBJECT | B_OBJECT | X_OBJECT | D_OBJECT | L_OBJECT )
U_OBJECT         = "U" "x" ESCAPED_URI "X"
N_OBJECT         = "N" "x" LEXICAL_FORM "X"
S_OBJECT         = "S" "x" LEXICAL_FORM "X" ( "L" LANG )?
B_OBJECT         = "B" "x" LEXICAL_FORM "X"
X_OBJECT         = "X" "x" LEXICAL_FORM "X"
D_OBJECT         = "D" "x" LEXICAL_FORM "X"
L_OBJECT         = "L" "x" "d" DATATYPE_URI "D" LEXICAL_FORM "X"
REIFICATION      = "R" PROPERTY_VALUE
DATATYPE_URI     = ESCAPED_URI
URI              = <a non-escaped, literal URI>
ESCAPED_URI      = <a Facet encoded URI>
LEXICAL_FORM     = <a Facet encoded lexical representation of the property value>
LANG             = <a Facet encoded ISO conformant language code, with optional dialect, all lowercase>
```

```

WHITESPACE      = <a whitespace character>

where

"U" = URI          (any valid URI)
"S" = String       (any valid Unicode string)
"N" = Number       (any valid lexical form of xsd:decimal)
"D" = Date(Time)   (any valid lexical form of xsd:date or xsd:dateTime)
"B" = Boolean      (any valid lexical form of xsd:boolean)
"X" = XML          (any valid lexical form of rdfs:XMLLiteral)
"L" = Literal Value (any valid Unicode string)

```

Sequences of more than one reification are ordered alphabetically first by the URI of the predicate and then, for the same predicate, by the lexical form of the object.

## Facet Encoding

Facet encoding employs the following reserved characters:

```
e E x X d d R L
```

Any character in the input which is either a reserved character, or not an alphanumerical character (a-zA-Z0-9) is escaped as

```
"e" DECIMAL_VALUE "E"
```

The resulting encoding string consists entirely of alphanumeric characters.

Thus, the non-alphanumeric characters in the URI "http://sw.nokia.com/FN-1/published" are escaped to become

```
http58Ee47Ee47Esw46Enokiae46Ecome47EFNe45E1e47Epublishe101Ee100E
```

Similarly, the string "A very cool ringtone" becomes

```
Ae32Eve101Erye32Ecoole32Eringtone101E
```

Thus, the above RDF statement about the ringtone would be represented as an RDF Facet Description as

```
<http://example.org/coolRingtone>
xhttp58Ee47Ee47Epurle46Eorge47Ee100Ece47Ee101E1e101Eme101Entse47E1e46E1e47Etitle101EXSxAe32Eve101Erye32Ecoole32Erington
e101EX
```

In the index, the property+value (predicate+object) portion can be treated as a single keyword or "facet" associated with the target (subject). Since the encoding used is fully bidirectional, one can later obtain the actual URIs, lexical forms, etc. as required for comparisons or for e.g. mapping the facets back to RDF/XML.

Any web search engine which can accept strings comprised of alphanumeric characters as keywords (which should include most every web search engine ever used) will be able to accept RDF Facets as index keywords (though some search engines may not support arbitrarily long keywords, which could preclude the use of overly long facets with some search engines). Thus the keyword

```
xhttp58Ee47Ee47Epurle46Eorge47Ee100Ece47Ee101E1e101Eme101Entse47E1e46E1e47Etitle101EXSxAe32Eve101Erye32Ecoole32Erington
e101EX
```

representing the property+value pair for a DC title of "A very cool ringtone" can be associated with the resource in the same way as a simple token keyword occurring in some indexed textual content, and comparison is performed using a simple string equality test the same as for any keyword.

Furthermore, the length of the RDF derived keyword should not impose a substantial efficiency penalty for most search engines, as most employ hashing and similar methods to optimize keyword storage and comparison, thus even if thousands of resources would be qualified with a given RDF property+value facet, the complete facet string will typically be stored only once.

The following is an example of a facet encoding of a statement with reification for a resource classified for the topic 'java' with 30% relevance, i.e.

```
<http://example.org/someResource> <http://sw.nokia.com/FN-1/topic> <http://sw.nokia.com/FN-1/topic/java> .
_:s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement> .
_:s <http://www.w3.org/1999/02/22-rdf-syntax-ns#subject> <http://example.org/someResource> .
_:s <http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate> <http://sw.nokia.com/FN-1/topic> .
_:s <http://www.w3.org/1999/02/22-rdf-syntax-ns#object> <http://sw.nokia.com/FN-1/topic/java> .
_:s <http://sw.nokia.com/MARS-3/relevance> "30"^^<http://sw.nokia.com/MARS-3/Percentage> .
```

where the datatype `<http://sw.nokia.com/MARS-3/Percentage>` maps to a number ("N") value type, would be represented as an RDF Facet as

```
<http://example.org/someResource>
xhttp58Ee47Ee47Esw46Enokiae46Ecome47EFNe45E1e47EtopicXUxhttp58Ee47Ee47Esw46Enokiae46Ecome47EFNe45E1e47Etopic47Ejava
XRxhttp58Ee47Ee47Esw46Enokiae46Ecome47EMaE82ESe45E3e47Ere101E1e101Evance101EXN30X
```

During indexing, the above reified facet should be expanded into two facets inserted into the search index, one with and one without the reification, to facilitate more efficient identification of resources with the specified facet where reification qualification is not relevant:

```
IF
<http://example.com/someResource>
xhttp58Ee47Ee47Esw46Enokiae46Ecome47EFNe45E1e47EtopicXUxhttp58Ee47Ee47Esw46Enokiae46Ecome47EFNe45E1e47Etopic47Ejava
XRxhttp58Ee47Ee47Esw46Enokiae46Ecome47EMaE82ESe45E3e47Ere101E1e101Evance101EXN30X

THEN
<http://example.com/someResource>
xhttp58Ee47Ee47Esw46Enokiae46Ecome47EFNe45E1e47EtopicXUxhttp58Ee47Ee47Esw46Enokiae46Ecome47EFNe45E1e47Etopic47Ejava
XRxhttp58Ee47Ee47Esw46Enokiae46Ecome47EMaE82ESe45E3e47Ere101E1e101Evance101EXN30X
xhttp58Ee47Ee47Esw46Enokiae46Ecome47EFNe45E1e47EtopicXUxhttp58Ee47Ee47Esw46Enokiae46Ecome47EFNe45E1e47Etopic47Ejava
X
```

Likewise, facets with language qualified string values can be simplified to their non-qualified forms:

```
IF
<http://example.com/someResource>
xhttp58Ee47Ee47Eepurle46Eorge47Ee100Ece47Ee101E1e101Eme101Entse47E1e46E1e47Etitle101EXSxAe32Eve101Erye32Ecoole32Erington
e101EXLen

THEN
<http://example.com/someResource>
xhttp58Ee47Ee47Eepurle46Eorge47Ee100Ece47Ee101E1e101Eme101Entse47E1e46E1e47Etitle101EXSxAe32Eve101Erye32Ecoole32Erington
e101EXLen
xhttp58Ee47Ee47Eepurle46Eorge47Ee100Ece47Ee101E1e101Eme101Entse47E1e46E1e47Etitle101EXSxAe32Eve101Erye32Ecoole32Erington
e101EX
```

I.e. to get the simplified facet of any reification or language qualified facet, simply strip off the end of the facet from the first "R" or "L" character onwards. E.g. 's/[RL].\*\$//'.

## RDF Facet Examples

To see a comprehensive body of knowledge expressed as RDF Facets, try the following query URI:

```
http://sw.nokia.com/new?since=one-month-ago&format=application/rdf-facets
```

## RDF Query Facet Syntax

When it comes time to execute an RDF-based query, we can construct a variant of the index facets which can be efficiently compared. These are intended to be compared against the database of index facets (or some internal representation specific to the particular search engine), with the goal being the identification of the subjects having the matched index facets.

The syntax for RDF Query Facets is:

```

QUERY          = QUERY_FACET ( WHITESPACE+ QUERY_FACET )*
QUERY_FACET    = QUALIFIED_VALUE REIFICATION*
QUALIFIED_VALUE = ( "ANY" | "NO" ) PREDICATE
QUALIFIED_VALUE = U_OPERATOR? PREDICATE U_OBJECT
QUALIFIED_VALUE = N_OPERATOR? PREDICATE N_OBJECT
QUALIFIED_VALUE = S_OPERATOR? PREDICATE S_OBJECT
QUALIFIED_VALUE = B_OPERATOR? PREDICATE B_OBJECT
QUALIFIED_VALUE = X_OPERATOR? PREDICATE X_OBJECT
QUALIFIED_VALUE = D_OPERATOR? PREDICATE D_OBJECT
QUALIFIED_VALUE = L_OPERATOR? PREDICATE L_OBJECT
PREDICATE      = "x" ESCAPED_URI "X"
REIFICATION    = "R" QUALIFIED_VALUE
U_OBJECT       = "U" "x" ESCAPED_URI "X"
N_OBJECT       = "N" "x" LEXICAL_FORM "X"
S_OBJECT       = "S" "x" LEXICAL_FORM "X" ( "L" LANG )?
B_OBJECT       = "B" "x" LEXICAL_FORM "X"
X_OBJECT       = "X" "x" LEXICAL_FORM "X"
D_OBJECT       = "D" "x" LEXICAL_FORM "X"
D_OBJECT       = MILESTONE
L_OBJECT       = "L" "x" "d" DATATYPE_URI "D" LEXICAL_FORM "X"
U_OPERATOR     = ( "NE" )
N_OPERATOR     = ( "NE" | "LT" | "GT" | "LE" | "GE" )
S_OPERATOR     = ( "NE" | "LT" | "GT" | "LE" | "GE" )
B_OPERATOR     = ( "NE" )
X_OPERATOR     = ( "NE" )
D_OPERATOR     = ( "NE" | "LT" | "GT" | "LE" | "GE" )
L_OPERATOR     = ( "NE" )
MILESTONE      = ( "HOUR" | "DAY" | "WEEK" | "MONTH" | "YEAR" )
DATATYPE_URI   = ESCAPED_URI
ESCAPED_URI    = <a Facet encoded URI>
LEXICAL_FORM   = <a Facet encoded lexical representation of value>
WHITESPACE     = <a whitespace character>

where

"ANY"          (resource has any (some) value defined for property)
"NO"           (resource has no value defined for property)
"HOUR"         (date-time equal to one hour ago from present)
"DAY"          (date-time equal to one day ago from present)
"WEEK"         (date-time equal to one week ago from present)
"MONTH"        (date-time equal to one month ago from present)
"YEAR"         (date-time equal to one year ago from present)

and

"U" = URI          (any valid URI)
"S" = String       (any valid Unicode string)
"N" = Number       (any valid lexical form of xsd:decimal)
"D" = Date(Time)   (any valid lexical form of xsd:date or xsd:dateTime)
"B" = Boolean      (any valid lexical form of xsd:boolean)
"X" = XML          (any valid lexical form of rdfs:XMLLiteral)
"L" = Literal Value (any valid Unicode string)

```

Sequences of more than one reification are ordered alphabetically first by the URI of the predicate and then, for the same predicate, by the lexical form of the object.

The absence of any operator at the beginning of the query facet indicates a test for equality. Note that this equality query facet form is equivalent to the predicate+object portion of an index facet, allowing for comparison using a simple string equality test.

Furthermore, one can easily differentiate equality query facets from other query facets by checking the first character of the query facet. If it is "x" then the query facet is a test for equality, otherwise, it involves some other value comparison operation.

A simple search engine could support only equality tests, and still benefit greatly from employing this methodology to obtain string encoded keyword facets derived from RDF property+value pairs.

Note that this facet based query model, is "flat" and resource centric, reflecting the general nature of typical web search engines. As such, it does not provide for structured values (where the value is a blank node with additional properties) nor does it provide for complex queries where more than one resource is described by the query (e.g. find all documents who's author's first name is "Bob").

It also defines a query as a simple set of query facets, and it is not specified how query facets might be otherwise combined in structured boolean expressions or whether all or some of a set of query facets must match (i.e. complete vs. best match). Different search engines will provide different levels of functionality in this regard. This document aims to nail down the common and most fundamental bits which should be relevant to most web search engines.

The following is a query facet corresponding to the query "match all targets with a FN publication date later than (greater than) 2004-01-01":

```
GTxhttp58Ee47Ee47EswE46Enokiae46Ecome47EFNe45E1e47Epublishe101Ee100EXDx2004e45E01e45E01X
```

which would match the index facet

```
<http://example.org/myRingtone>
xhttp58Ee47Ee47EswE46Enokiae46Ecome47EFNe45E1e47Epublishe101Ee100EXDx2004e45E03e45E31X
```

by ultimately being evaluated within the search engine as a comparison of the dates, e.g.

```
? datetime("2004-03-31") > datetime("2004-01-01")
```

thereby identifying the target <http://example.org/myRingtone>, etc.

An example query facet which identifies all resources relating to the FN topic 'java' with a relevance greater than or equal to 20 would be:

```
xhttp58Ee47Ee47EswE46Enokiae46Ecome47EFNe45E1e47EtopicXUxhttp58Ee47Ee47EswE46Enokiae46Ecome47EFNe45E1e47Etopice47Ejava
XRGExhttp58Ee47Ee47EswE46Enokiae46Ecome47EMAe82ESe45E3e47Ere101E1e101Evance101EXNx20X
```

A query facet which identifies all resources having any (some) FN type definition (i.e. all FN resources) would be:

```
ANYxhttp58Ee47Ee47EswE46Enokiae46Ecome47EFNe45E1e47Etype101EX
```

A conjunctive query (where all query facets must match) which identifies all resources having any FN type definition but having no FN topic definition would be:

```
ANYxhttp58Ee47Ee47EswE46Enokiae46Ecome47EFNe45E1e47Etype101EX
NOxhttp58Ee47Ee47EswE46Enokiae46Ecome47EFNe45E1e47EtopicX
```

## Language and Reification Qualifications

If LANG is specified in the query, then string values must be specified for language and the language either must match the specified LANG exactly, or else, if the LANG value does not include a dialect, must match the language prefix. I.e.

Query		Index
-----		-----
xabcX	matches	xabcX
xabcX	matches	xabcXLen
xabcXLen	matches	xabcXLen
xabcXLen	matches	xabcXLen95Eus
xabcXLen95Eus	matches	xabcXLen95Eus
but		
xabcXLen95Eus	does not match	xabcXLen95Euk

```

xabcXLen95Eus does not match xabcXLen
xabcXLen      does not match xabcX

```

If reifications are specified in the query, then each reification in the query facet must match an explicit reification in the index facet. Reifications in the index facet not relevant to the query facet can be ignored.

Thus, the query facet

```
xhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47EtopicXxhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47Etopice47EjavaX
```

matches the facet descriptions

```

<http://example.org/someResource>
xhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47EtopicXxhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47Etopice47EjavaX
Rxhttp58Ee47Ee47Eswe46Enokiae46Ecome47EMaE82ESe45E3e47Ere101E1e101Evance101EXNx30X

```

and

```

<http://example.org/someResource>
xhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47EtopicXxhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47Etopice47EjavaX

```

but the query facet

```

xhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47EtopicXxhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47Etopice47EjavaX
Rxhttp58Ee47Ee47Eswe46Enokiae46Ecome47EMaE82ESe45E3e47Ere101E1e101Evance101EXNx30X

```

does not match the description

```

<http://example.org/someResource>
xhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47EtopicXxhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47Etopice47EjavaX

```

The fundamental rule is that equal matches equal and less precise/detailed matches more precise/detailed.

## Query Analysis of User Input Keywords

Free-form, user input, natural language based keywords can be associated with particular facets, as a means to map from user input to precise metadata classifications, as well as a means to map from precise queries to federated search engine interfaces which have no RDF or RDF Facet support.

E.g., we could associate the input keywords "games" and "gaming" with the facet

```
xhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47EtopicXxhttp58Ee47Ee47Eswe46Enokiae46Ecome47EFNe45E1e47Etopice47Egame101EsX
```

corresponding to the property+value pair for the FN 'topic' property and the 'games' topic value. When a user inputs into a basic search field either keyword "games" or "gaming", rather than searching for those actual words in the textual content of resources, we instead interpret the query to mean they are looking for resources relating to the precise topic 'games', employing the more precise facet, thereby providing more precise and useful search results.

This mapping between keywords and metadata facets is fully bidirectional.

I.e., a query broker could use this mapping to generate keyword-based searches to external search engines from query facets. Thus, the above query facet could be mapped to a Google search sub-pattern 'games OR gaming', joining all keywords associated with the facet disjunctively.

One of the greatest benefits to this approach is that it allows most of the knowledge to be maintained in a declarative fashion, and also simplifies the query broker's mapping task to one strictly concerned

with syntax (how to express conjunctive keyword searches) -- which reduces the overall implementation complexity and also should significantly enhance runtime efficiency.

## Inference and Run-Time Search Index Generation

When generating/updating the run-time search index based on the RDF index facets, inference/reasoning should be employed to produce a maximal facet set for any given resource. Thus, if we know, based on our ontologies and their defined relationships, that e.g. any resource classified for topic "X" also pertains to topic "Y", we will generate the explicit facet for both topics "X" and "Y" as well.

When someone then searches for resources pertaining to topic "Y", they will find the above mentioned resource, even though in the knowledge management side, it is only explicitly classified for topic "X", because the run-time search engine will maintain a maximal view of that resources characteristics, both explicitly and implicit.

This approach allows the indexer/search engine to be as simple as possible (not having to provide support for keyword aliases, taxonomies, etc.) and thus also be as efficient as possible (the two tend to go hand in hand) since the bulk of the complexity is confined to the indexing process and not to run-time execution of queries.

E.g. to see the above example facet set with all inferable facets included, try the following

```
http://sw.nokia.com/new?since=one-month-ago&format=facets&inference=include
```

You'll note that while this query takes a little bit longer to execute (since the server has to reason about all the ontological relations) it produces a much more detailed set of facets, including, e.g. all of the Dublin Core and RSS vocabulary equivalents to the FN vocabulary terms, as well as all subtype/subtopic inheritance classifications, etc.

And since this inference/reasoning is done during the index generation process, the ontological complexity does not directly impact the run-time efficiency of the publicly-accessible search service.