Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

X-Values

Typed Data Literals for the Semantic Web and Beyond

This document outlines a proposed methodology, named 'X-Values', which employs the definition of a URI scheme and its RDF application for identifying globally valid, typed data values. Such identifiers include both an explicitly defined data type and the data value itself, encoded as a URI and thus may serve as a portable and generic point of intersection between all systems and applications interchanging and utilizing such values.

The specification of this methodology is considered to be incomplete, and not yet suitable for implementation and deployment as-is. It is intended to serve only as the basis for discussion, and ultimately as the foundation of a more rigorous and formal specification.

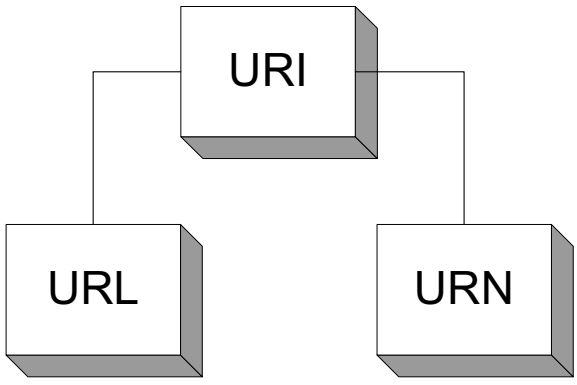
1. DESCRIPTION

The proposed methdology is comprised of the following components:

A. A URI scheme for explicitly representing data literals according to data type
B. A set of predefined data types; specifically including a data type providing a means of URI reification
C. A mechanism for defining ad-hoc yet globally unique custom data types
D. An ontology, defined in RDF, for specifying the essential semantics of and constraints on values which are members of a given data type; specifically providing for the definition of valid lexical forms in a general and system portable fashion
E. An explicit equivalence relation between particular variant RDF graph representations of what is considered to be equivalent knowledge

1.1 Conceptual Basis: A (Partial) Taxonomy of URI Schemes

There presently are defined two sub-classes of URI: Uniform Resource Locators (URLs) and Uniform Resource Names (URN):

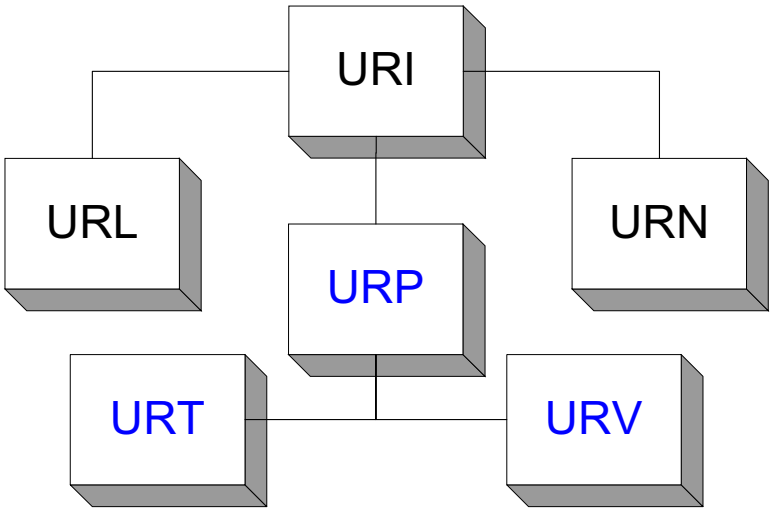


Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

URLs are perhaps the most widely understood and certainly most widely used class of URIs; therefore no additional discussion is provided here. URNs are a bit less widely understood than URLs and therefore a basic summary is provided below.

In addition to URLs and URNs, we can define an additional class of URI: Uniform Resource Primitives (URP), which is itself divided into two sub-classes: Uniform Resource Terms (URT), and Uniform Resource Values (URV):



1.1.1 Uniform Resource Name (URN)

A URN is used as location independent identifier for a retrievable web resource.

URNs are similar to URLs in that they are expected to resolve to a web resource; however, in the case of URLs, both the creating authority and resolution agency are specified in the URI form and are identical; but with URNs only the creating authority is specified in the URI and the resolution agency is left unspecified and may be variable and context dependent.

Each URN scheme defines a virtual system by which web resources may be accessed via keys corresponding to the instances of that URN scheme. The realization of that system may consist of the collaboration of a large number of agencies and (possibly redundant) physical systems. Nevertheless, they collectively constitute a single system with a single set of indirect access keys.

The creating authority of a URN may be mnemonic (e.g. tag) or non-mnemonic (e.g. uuid, isbn, doi, bitprint, etc).

All URNs are defined by an identifiable authority which is either equivalent to the authority defining the scheme or is discernable from the URI itself. All URNs are expected to be defined by a centrally managed authority or an authorized proxy in order to ensure uniqueness and integrity of instances of the scheme for any given generating authority.

Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

Examples of existing or potential URN schemes: tag, uuid, isbn, doi, bitprint, ean, essentially any controlled set of location-independent indirect resource identifiers.

1.1.2 Uniform Resource Primitive (URP)

URPs contstitute a class of URIs which we may describe as "fully resolved" or "non-dereferencable" such that they are self-contained and do not represent any other resource – hence the term ‘primitive’. The entire resource is embodied in the actual URI form. In contrast to most other URI schemes, which serve as mechanisms of indirection, their purpose is primarily for attribution or classification of other resources.

One may make statements about those resources (e.g. via RDF), as one may do about any resource, but they do not resolve or dereference to any other resource. URPs constitute “WYSIWYG” URIs. I.e. “what you see (in the URI) is what you get”. This quality should become clear following the discussion of URTs and URVs below.

1.1.3 Uniform Resource Term (URT)

A URT scheme represents a finite, controlled and explicitly enumerated vocabulary of terms, possibly constituting and organized as a taxonomy.

All URTs are defined by the same authority defining the scheme, or a proxy agency authorized by that authority, and the creating authority is (directly or indirectly) the same as for the scheme itself.

Examples of existing or potential URT schemes: IETF/ISO standard language and country codes, TGN, Psyc, WordNet, and all vocabularies for essentially all abstract ontologies.

1.1.4 Uniform Resource Value (URV)

A URV scheme is a lexically defined data type for literal data values (e.g. the X-Value URV scheme defined below as the primary component of this methodology).

URVs have the unique quality that they may be defined by agents other than the authority defining the scheme, so long as they are defined in accordance with the lexical constraints defined for the scheme; and unlike all other URI schemes, the creating authority is not discernable from the URI itself.

Examples of existing or potential URV schemes: IETF/ISO date and time formats, X-Value data types, Unicode code points.

The key difference between URTs and URVs is that all values of a URT are defined (directly or indirectly) by a single, central authority, whereas URVs may be defined by anyone, so long as they conform to the lexical constraints defined for the scheme.

Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

1.2 URV Scheme: X-Types

[Note: This discription borrows existing definitions from RFC1738 (URL Schemes); however a formal specification of the URI scheme for registration with IANA will make such definitions explicitly to avoid nonessential dependencies to other specifications]

A URI according to this scheme can take one of two alternate forms:

a. A “core” data type, defined as part of the definition of the URI scheme itself.
b. A “custom” data type, defined within the context of an internet authority.

1.2.1 Core Data Type URI Forms

Core data type URIs consist of three parts:

1. A fixed prefix consisting of the lowercase string 'x' followed by a colon ‘:’
2. A predefined data type name matching the regular expression pattern /[a-z][_a-z]*/ followed by a colon ‘:’
3. A URL encoded data value (with relevant characters escaped as specified by RFC1738)

Note that many (likely most) data types will not require any escaped characters as they are by definition URL encoded (or rather do not permit characters which must be escaped for inclusion in a URI.

E.g.

```
x:int:5
x:float:1.247882899321004318e5
x:char:W
x:str:Here%20is%20a%20string%20with%20spaces.
x:date:2001-09-28
x:time:2001-09-28T11:32:09Z
x:mm:32cm
x:bool:true
x:uri:http://www.nokia.com
x:uri:x:int:5
```

Predefined core data types ideally will be compatible with, and conformant to, as broad a range of existing standards as possible, where relevant, including those defined by POSIX, ANSI, ISO, W3C, IETF, IEEE, NISO, etc.)

[Note: the final set of core data types is yet to be defined. The examples provided are only intended to be illustrative of possible core data types.]

Examples of possible core data types and their URI component names are:

Name	***Type***
str	string
int	integer
float	floating point number
rank	ordered tree index (c.f. Metia Framework definition)
char	character

Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

date	date
time	date and time
name	name token (c.f. SGML/XML NAME type)
mm	measured magnitude (per explicit unit of measure)
regex	regular expression pattern
uri	uniform resource identifier

1.2.2 Custom Data Type URI Forms

Custom data type URIs consist of four parts:

1. A fixed prefix consisting of the lowercase string 'x' followed by a colon ':'
2. An authority identifier conforming to the "<user>@<host>" template subset as defined in RFC1738 for URL common scheme syntax, including optionality of the "<user>@" portion.
3. A custom data type name matching the regular expression pattern /[a-z][_a-z]*/ followed by a colon ':'
4. A URL encoded data value (with relevant data characters escaped as specified by the HTTP URI specification)

E.g.

x:abc.com:xyzcoord:28.992.143
x:john.doe@foo.org:foohash:82884a11f2e83

1.2.3 Authority over Type, not Values: Lexical Basis for Definition

The specification of data types are intended to be based solely on a lexical representation defining a presumed infinte set of values corresponding to that data type where values conforming to that data type may be defined by anyone, not just the authority that defines the data type. Thus, controlled vocabularies, taxonomies, enumerations of term sets and the like may not be defined using this scheme as there is no central authority which controls all values which may be defined according to a given data type.

1.2.4 Standardized Reification of URIs

The set of core data types will include a data type for URIs in general, providing a standardized means by which URIs may be reified, allowing RDF statements to be made about the URIs themselves, rather than about the resources which they denote, or in the case of WYSIWYG URIs (URPs) the implicit resources which they themselves constitute.

1.2.5 Not an Indirect Identifier

The following constraint is defined with regards to this URI scheme:

> URIs defined according to this scheme identify generic values of a particular data type and those values are not intended to serve as identifiers for other resources, though they may be used to classify or qualify other resources or serve as the values of properties of other resources.

Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

Again, because there is no central authority controlling which values can be defined and used for a given data type – apart from lexical constraints – there is no means by which one can ensure the global uniqueness of any value as an identifier for a single resource.

This does not mean that URIs defined according to this scheme do not themselves identify explicit resources, only that those resources represent the data values themselves, and thus are expected to be at best semantically poor. I.e. an X-Value URV is a WYSIWYG URI.

1.2.6 Summary

In summary, this URV scheme provides:

- A unique yet consistent identifier for every typed data literal value (for which a data type can be defined according to this scheme), which spans the entire global internet/intranet/extranet scope
- The ability for any agency having a valid web-based authority identifier to define additional data types without recourse to any centralized registration authority
- A set of core data types that are meaningful to a very broad number of systems and applications
- A means to define the semantics and lexical form of any data type explicitly by means of RDF, such that it can be determined at run-time by applications (see below)
- An identifier having a format that is relatively easy for users to type in, if ever necessary (particularly for the most fundamental data types which do not require URL encoded escaped characters), and is maximally mnemonic and recognizable to humans
- A standardized, generalized, consistent means of URI reification

Although this solution could possibly be percieved as being able meet all known requirements for URNs in general, by defining the equivalent of URN Namespaces as core or custom data types, this is not the case, as instances of identifiers defined by this scheme are only constrained by lexical form and furthermore may not act as identifiers for other resources. E.g. any statements made using such identifiers as the subject of an RDF statement modify the data value, not some other resource identified by that value. I.e., they are representations of data value resources, not identifiers of other resources.

1.3 Data Type Ontology

Associated with the above defined URI scheme, and included as part of the X-Values methodology, is an ontology of properties and relations defined in RDF (Resource Description Framework), and grounded in the XML namespace “x:” which serves to explicitly define the semantics and lexical constraints of a particular data type.

[Note: the final ontology is yet to be defined. The examples provided are are only intended to be illustrative of possible properties.]

Examples of possible properties and relations defined for a particular data type could be:

Property	***Purpose***
pattern	inclusive regular expression pattern defining acceptable lexical form
xpattern	exclusive regular expression pattern defining unacceptable lexical form
subType	defines relations between data types (a type hierarchy)

Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

equivalentTo	specifies that two data types or values should be treated as equal
approximateTo	specifies that two data types or values should be treated as approximately equal, but not precisely equal (intepretation of precision left up to application)
definedBy	specifies where a given data type or value is formally defined
conformsTo	specifies a standard to which the data type or value conforms

There is no restriction against there existing multiple defined pattern and xpattern properties, in which case a value is considered to be valid if it matches any of the defined patterns and does not match any of the defined xpatterns.

The ability to define cooperative inclusive and exclusive patterns serves to simplify the definition of the valid lexical form of certain data types (such as dates), which have proven to be easier and/or more efficient to define by exclusions following more generalized, less restrictive inclusive patterns. [1]

Although it is technically feasible to define an enumeration, such as for a controlled term set according to one or more regular expressions, to enumerate verbatim all possible values for a data type is considered contrary to the purpose of this scheme and is prohibited. Data types defined for this scheme are expected (though not absolutely required) to accommodate an infinite number of values.

It is expected that (wherever feasible) that the lexical constraints defined for a given data type do not allow semantically vacuous variant forms, e.g. “x:int:5” and “x:int:000005” as this will diminish or defeat the benefit of all equivalent values having the same URI representation and hence constitute a common point of intersection within any RDF graph utilizing such URIs.

1.4 Equivalence Relations between RDF Graph Variants

The following three RDF graphs are defined by this methodology to have identical interpretation with regards to data typing and constitute the same fundamental knowledge, and thus should receive similar if not identitical treatment by any application utilizing X-Value defined data typed values:

1. Explicit X-Value encoded resource property value.

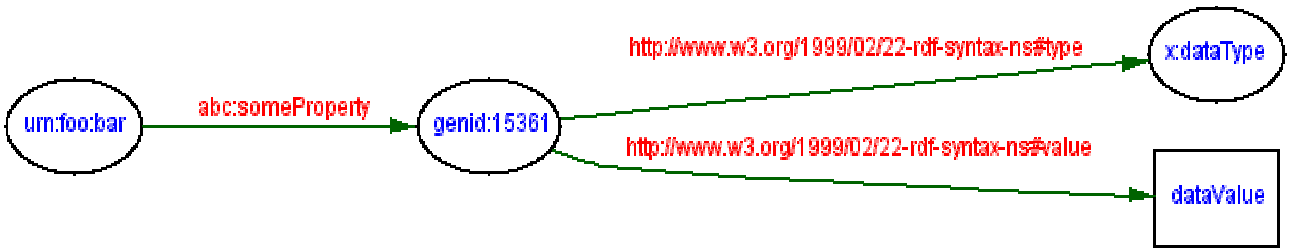
```
<rdf:Description rdf:about="urn:foo:bar">
   <abc:someProperty rdf:resource="x:dataType:dataValue"/>
</rdf:Description>
```

Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

2. X-Value typed anonymous node with literal value.

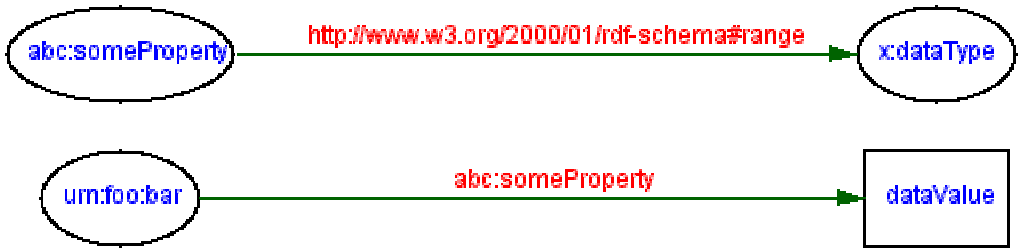
```
<rdf:Description rdf:about="urn:foo:bar">
   <abc:someProperty>
      <rdf:Description>
         <rdf:type rdf:resource="x:dataType"/>
         <rdf:value>dataValue</rdf:value>
      </rdf:Description>
   </abc:someProperty>
</rdf:Description>
```



3. "Locally untyped" literal property value with property having an X-Value type range.

```
<rdf:Description rdf:about="urn:foo:bar">
   <abc:someProperty>dataValue</abc:someProperty>
</rdf:Description>

<rdf:Description rdf:about="abc:someProperty">
   <rdfs:range rdf:resource="x:dataType"/>
</rdf:Description>
```



Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

2. DISCUSSION

This methodology enables the representation of data types literals (e.g. integers, strings, dates, floating point values, measurements, etc.) as URIs which provides a system-independent, global, portable, and consistent representation across all applications which are URI aware but may not have access to XML Schema and/or RDF Schema defined typing definitions.

It further enables (though does not require) the elimination of literal values in RDF and all of the machinery necessary to define and maintain the distinction between resource objects of statements versus literal objects of statements.

Within the context of internet applications interchanging typed data literals, such as metadata interchange, being able to define data values as URVs, per the methodology in question, helps to ensure that:

a) every value is associated with an explicit data type (not just an otherwise untyped string)

This relieves the burden of interpretation on any application, since the application will know exactly what type of data the value is supposed to represent (even if it is not familiar with or able to deal with that particular data type). I.e. there is no ambiguity whether the value string "135.68" represents a floating point number or an x/y coordinate, etc.

b) every value could be lexically validated according to that data type

Since each value is defined within the context of a data type, and knowledge about that data type can be defined in a standardized fashion, including the lexical (format) characteristics of legitmate values of that data type, an application is able to easily check whether the value specified is in fact a valid lexical representation for a value of that data type. E.g. "x:int:foo" would (we hope) be rejected, since 'foo' is not a valid lexical form for an integer.

c) every instance of the same value would be treated as identical

Thus, for example, in any URI based knowledge base (RDF, XTM, etc), one could perform more efficient data mining such as "what properties share a common date, irrespective of the intepretation of that date" e.g. when looking for knowledge about events happening on the same day as some other significant event, etc. All statements including a given date encoded as a URV will intersect in the same exact resource. No need to scan strings to find all that match a given pattern.

d) every system using data types corresponding to those defined in such a global standard, such as described by this methodology, can safely and efficiently map between the compact URV representations and their own internal representations or between alternate RDF graph representations, with minimal or no recourse to lexical analysis of the literal forms or consultation of ontologies or object models.

The set of predefined data types (either core or custom) thus serves as a ‘lingua franca’ of typed data literals for all IM systems globally which are URI aware.

Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

2.1 Weak Data Typing versus Strong Data Typing

The interpretation of RDF literals can be either weakly typed or strongly typed, based on whether data types are defined for each value individually (weak typing) or defined indirectly by associating a type with the property for which it serves as the value (strong typing).

One can consider literals for which no data type is defined, either by some schema or locally for each value occurrence as “untyped”, or at best having an implicit type corresponding to an uninterepted sequence of Unicode values (or an implied default RDF Schema type ‘Resource’). Note that RDF permits a combination of strongly typed, weakly typed, and untyped literal values in the same knowledge base. It is thus ultimately up to the end application to ensure the validity of any given value for any particular use.

Following the weakly typed approach, the most common method of associating literal values with their data types is by means of qualified anonymous nodes, where an anonymous resource node serves as a proxy value for the literal and points to the literal and one or more types. (c.f. “Poor Man’s Structured Values” as defined by DCMI [2]). Although very simple and flexible in principle, this method can be cumbersome to use in practice and can result in a rather verbose representation, both in serialized and graph form.

The X-Values methodology primarily supports the weak data typing approach, and can be seen as providing a more compact and generalized alternative to qualified anonymous nodes for associating data type information with values, and which facilitates broader interchange and interoperability between a larger range of web applications – not merely those based upon or which support RDF and RDF Schema.

The X-Values methodology also supports the strong data typing approach as well, insofar as validation of data values is concerned, as it provides for an explicit definition of valid lexical forms based on regular expressions which is sufficient for many, if not most, commonly used “primitive” data types and which has a far lower implementational burden than other alternatives. Thus, even if data types are defined by means of RDF Schema range constraints, one can validate literals according to the patterns defined for each data type using the X-Values ontology. One need not resort to a native data type solution beyond being able to test regular expressions against a literal string.

2.2 Minimal Centralized Control and Zero URI Management Overhead

Although a core set of data types will be defined by this URV scheme itself, any person, group, business, organization etc. with a web identity corresponding to an ‘authority’ as defined by this URV scheme can define their own data types and anyone can then create and interchange instances of those data types without recourse to any centralized registration or management agency. This achieves an optimal balance between reliability and flexibility with regards to the creation and management of URIs; allowing users to themselves decide which data type defining authorities they wish to trust; either in the creation of their own URIs or in the acceptance and interpretation of other’s.

It is also conceivable, even expected, that other URV schemes will be defined in addition to this one, further broadening the choice offered to users for global, portable, explicit URI encoded typed data literals.

Nokia Research Center
Software Technology Laboratory
Agent Technology Group

Patrick Stickler
patrick.stickler@nokia.com

References:

[1] "Regular expressions for checking dates", Eric Howland and David Niergarth, Markup Languages: Theory and Practice, ISSN 1099-6621, Volume 2, Issue 2 (Spring 2000), pages 126-132

[2] http://dublincore.org/documents/2001/08/29/dcq-rdf-xml/

