Semantic Web Publishing using Named Graphs

Jeremy J. Carroll[1], Christian Bizer[2], Patrick Hayes[3], and Patrick Stickler[4]

[1] Hewlett-Packard Labs,Bristol, UK
[2] Freie Universität Berlin, Germany
[3] IHMC, Florida, USA
[4] Nokia, Tampere, Finland

Abstract. The Semantic Web consists of many RDF graphs nameable by URIs. This paper extends the syntax and semantics of RDF to cover such Named Graphs. This enables RDF statements that describe graphs, which is beneficial in many Semantic Web application areas. In this paper, we explore the application area of Semantic Web publishing: Named Graphs allow publishers to communicate assertional intent, and to sign their graphs; information consumers can evaluate specific graphs using task-specific trust policies, and act on information from those Named Graphs that they accept. Graphs are trusted depending on: their content; information about the graph; and the task the user is performing. The extension of RDF to Named Graphs provides a formally defined framework to be a foundation for the Semantic Web trust layer.

1 Introduction

A simplified view of the Semantic Web is a collection of web retrievable RDF documents, each containing an RDF graph. The RDF Recommendation [4, 11, 19, 23], explains the meaning of any one graph, and how to merge a set of graphs into one, but does not provide suitable mechanisms for talking about graphs or relations between graphs. The ability to express metainformation about graphs is required for:

Data syndication systems need to keep track of provenance information, and provenance chains.
Restricting information usage Information providers might want to attach information about intellectual property rights or their privacy preferences to graphs in order to restrict the usage of published information [15, 25].
Access control A triple store may wish to allow fine-grain access control, which appears as metadata concerning the graphs in the store [21].
Signing RDF graphs As discussed in [12], it is necessary to keep the graph that has been signed distinct from the signature, and other metadata concerning the signing, which may be kept in a second graph.
Expressing propositional attitudes such as modalities and beliefs [20].

RDF reification has well-known problems in addressing these use cases as previously discussed in [14]. To avoid these problems several authors propose quads [3, 16, 21, 24], consisting of an RDF triple and a further URIref or blank node or ID. The proposals vary widely in the semantic of the fourth element, using it to refer to information sources, to model IDs or statement IDs or more generally to ‘contexts’.

We propose a general and simple variation on RDF, using sets of *named* RDF graphs. A set of Named Graphs is a collection of RDF graphs, each one of which is named with a URIref. The name of a graph may occur either in the graph itself, in other graphs, or not at all. Graphs may share URIrefs but not blank nodes.

Named Graphs can be seen as a reformulation of quads in which the fourth element's distinct syntactic and semantic properties are clearly distinguished, and the relationship to RDF's triples, abstract syntax and semantics is clearer.

We describe how Named Graphs can be used for Semantic Web publishing, looking in particular on provenance tracking and how it interacts with the choices made by consumers of Semantic Web information about which information to trust.

2 Abstract Syntax and Semantics of Named Graphs

RDF syntax is based on a mathematical abstraction: an RDF graph is defined as a set of triples. These graphs are represented by documents which can be retrieved from URIs on the Web. Often these URIs are also used as a name for the graph, for example with an `owl:imports`. To avoid confusion between these two usages we distinguish between Named Graphs and the RDF graph that the Named Graph encodes or represents. A Named Graph is an entity with two functions *name* and *rdfgraph* defined on it which determine respectively its name, which is a URI , and the RDF graph that it encodes or represents. These functions assign a unique name and RDF graph to each Named Graph, but Named Graphs may have other properties; and named graphs may be concrete resources rather than set-theoretic abstractions. We follow the RDF convention whereby graphs which are equivalent in the sense of [23] - i.e. which differ only in the identity of their blank nodes - are considered to be identical. This has the consequence that blank nodes are considered to be internal to a graph, i.e. that two distinct RDF graphs do not have any blank nodes in common.

In more detail, we define a set of Named Graphs $\mathbf{N}$ to be a function from a set of URI references to a set of RDF graphs, i.e. a set of pairs $\langle N, G\rangle$ where G is an RDF graph.[5] Each pair $\mathit{ng} = (n, g) \in N$ is a Named Graph in $\mathbf{N}$, and we write $n = \mathit{name}(\mathit{ng})$ and $g = \mathit{rdfgraph}(\mathit{ng})$.

An RDF interpretation I (as in [19]) *conforms* with a set of Named Graphs $\mathbf{N}$ when:

For every Named Graph $\mathit{ng} \in \mathbf{N}$, we have $I(\mathit{name}(\mathit{ng})) = \mathit{ng}$

Note that the Named Graph itself, rather than the RDF graph it intuitively "names", is the denotation of the name. We consider the RDF graph to be related to the Named Graph in a way analogous to that in which a class extension is related to a class in RDFS. This 'intensional' (c.f. [19]) style of modelling allows for distinctions between several 'copies' of a single RDF graph and avoids pitfalls arising from accidental identification of similar Named Graphs. Note also that in any conforming interpretation, named graphs are denoted by their labelling URIs and hence are first-class elements of the universe of discourse on exactly the same basis as all other resources.

[5] We have removed the legacy constraint that a literal cannot be the subject of a triple.

As noted, we follow the notion of graph equivalence defined in RDF [23] by treating two RDF graphs which differ only in the identity of their blank nodes as being the same graph. A more explicit approach would take graph equivalence from [23] (i.e. a 1:1 mapping on blank nodes, a *renaming* function), and say that a *nameblanked* RDF graph is an equivalence class under this equivalence relation of replacing blank nodes by other blank nodes under some renaming. Then the *rdfgraph* of a Named Graph is a *nameblanked* RDF graph. We generally will ignore this complication.

2.1 RDF Reification

A 'reified statement' [19] is a single RDF statement described and identified by a URIreference. Within the framework of this paper, it is natural to think of this as a Named Graph containing a single triple, blurring the distinction between a (semantic) statement and a (syntactic) triple. This provides a useful connection with the traditional use of reification and a potential migration path.

2.2 Accepting Graphs

A set of Named Graphs $\mathbf{N}$ is not given a single formal meaning. Instead, the formal meaning depends on an additional set $A \subset \text{domain}(N)$. A identifies some of the graphs in the set as *accepted*. Thus there are $2^{|\text{domain}(N)|}$ different formal meanings associated with a set of Named Graphs, depending on the choice of A. The meaning of a set of accepted Named Graphs $\langle A, \mathbf{N} \rangle$ is given by taking the graph merge $\bigcup_{a \in A} N(a)$, and then interpreting that graph with the RDF semantics [19] (or an extension), subject to the additional constraint that all interpretations I conform with $\mathbf{N}$.

The choice of A reflects that the individual graphs in the set may have been provided by different people, and that the information consumers who use the Named Graphs make different choices as to which graphs to believe. Thus we do not provide one correct way to determine the 'correct' choice of A, but provide a vocabulary for information providers to express their intensions, and suggest techniques with which information consumers might come to their own choice of which graphs to accept.

3 Concrete Syntaxes and Query Languages

A concrete syntax for Named Graphs has to exhibit the name, the graph and the association between them. We offer three concrete syntaxes: TriX and RDF/XML both based on XML; and TriG as a compact plain text format.

The TriX[14] serialization is an XML format which corresponds fairly directly with the abstract syntax, allowing the effective use of generic XML tools such as XSLT, XQuery, while providing syntax extensibility using XSLT. TriX is defined with a short DTD, and also an XML Schema.

In this paper we use TriG as a compact and readable alternative to TriX. TriG is a variation of Turtle [5] which extends that notation by using '{' and '}' to group triples into multiple graphs, and to precede each by the name of that graph. The following TriG document contains two graphs. The first graph contains information about itself. The second graph refers to the first one, (namespace prefix definitions omitted).

```
:G1 { _:Monica ex:name "Monica Murphy" .
      _:Monica ex:email <mailto:monica@murphy.org> .
      :G1 pr:disallowedUsage pr:Marketing }

:G2 { :G1 ex:author :Chris .
      :G1 ex:date "2003-09-03"^^xsd:date }
```

Named Graphs are backward compatible with RDF. A collection of RDF/XML [4] documents on the Web map naturally into the abstract syntax, by using the first xml:base declaration in the document or the URL from which an RDF/XML file is retrieved as a name for the graph given by the RDF/XML file.

There are currently two query languages for Named Graphs: RDFQ [27] uses an RDF vocabulary to structure queries. TriQL [7] is a graph patterns based query language inspired by RDQL [26]. A prototypical implementation of TriX, TriG and TriQL is described in [9].

4 Semantic Web Publishing

One application area for Named Graphs is publishing information on the Semantic Web. This scenario implies two basic roles embodied by humans or their agents: Information providers and information consumers. Information providers publish information together with meta-information about its intended assertional status. Additionally, they might publish background information about themselves, e.g. their role in the application area. They may also decide to digitally sign the published information. Information providers have different levels of knowledge, and different intentions and different views of the world. Thus seen from the perspective of an information consumer, published graphs are claims by the information providers, rather than facts.

Different tasks require different levels of trust. Thus information consumers will use different trust policies to decide which graphs should be accepted and used within the specific application. These trust policies depend on the application area, the subjective preferences and past experiences of the information consumer and the trust relevant information available. A naive information consumer might for example decide to trust all graphs which have been explicitly asserted. This trust policy will achieve a high recall rate but is easily undermineable by information providers publishing false information. A more cautious consumer might require graphs to be signed and the signers to be known through a Web-of-Trust mechanism. This policy is harder to undermine, but also likely to exclude relevant information, published by unknown information providers.

4.1 Authorities, Authorization and Warrants

Information providers using RDF do not have any explicit way to express any intention concerning the truth-value of the information described in a graph; RDF does not provide for the expression of *propositional attitudes*. Information consumers may require this, however. Note that this is in addition to trust policies, and may be required in order to put such policies into operation. For example a simple policy could be: believe anything asserted by a trusted source. In order to apply this, it is necessary to have a clear record of what is *asserted* by the source. Not all information provided by a source need be asserted by that source. We propose here a vocabulary and a set of concepts designed to enable the uniform expression of such propositional attitudes using named graphs.

We take three basic ideas as primitive: that of an *authority*, a relationship of *authorizing*, and a *warrant*. An authority is a 'legal person'; that is, any legal or social entity which can perform acts and undertake obligations. Examples include adult humans, corporations and governments. The 'authorizing' relationship holds between an authority or authorities and a Named Graph, and means that the authority in some sense commits itself to the content expressed in the graph. Whether or not this relationship in fact holds may depend on many factors and may be detected in several ways (such as the Named Graph being published or digitally signed by the authority). Finally, a warrant is a resource which records a particular propositional stance or intention of an authority towards a graph. A warrant asserts (or denies or quotes) a Named Graph and is authorized by an authority. One can think of warrants as a way of reducing the multitude of possible relationships between authorities and graphs to a single one of authorization, and also as a way of separating questions of propositional attitude from issues of checking and recording authorizations. The separation of authority from intention also allows a single warrant to refer to several graphs, and for a warrant to record other properties such as publication or expiry date.

To describe the two aspects of a warrant we require vocabulary items: a property `swp:authority` (where `swp:` is a namespace for Semantic Web publishing) relating warrants to authorities, and another to describe the attitude of the authority to the graph being represented by the warrant. We will consider two such intentions expressed by the properties `swp:assertedBy` and `swp:quotedBy`. These take a named graph as a subject and a `swp:Warrant` as object; `swp:authority` takes a warrant as a subject and a `swp:Authority` as an object. Each warrant must have a unique authority, so `swp:authority` is an OWL functional property. Intuitively, `swp:assertedBy` means that the warrant records an endorsement or assertion that the graph is true, while `swp:quotedBy` means that the graph is being presented without any comment being made on its truth. This latter is particularly useful when republishing graphs as part of a syndication process, the original publisher may assert a news article, but the syndicator, acting as a common carrier, merely provides the graph as they found it, without making any commitment as to its truth. Warrants may also be signed, and the property `swp:signatureMethod` can be used to identify the signature technique.

4.2 Warrant Descriptions as Performatives

A warrant, as described above, is a social act. However, it is often useful to embody social acts with some record; for example a contract (which is a social act) may be embodied in a document, which is identified with that act, and is often signed. In this section, we introduce the notion of a *warrant graph*, which is a Named Graph describing a warrant, that is identified with the social act. Thus, this is a resource which is both a `swp:Warrant` and an `rdfg:Graph`. Consider a graph containing a description of a warrant of another Named Graph, such as:

```
{ :G2 swp:assertedBy _:w .
```

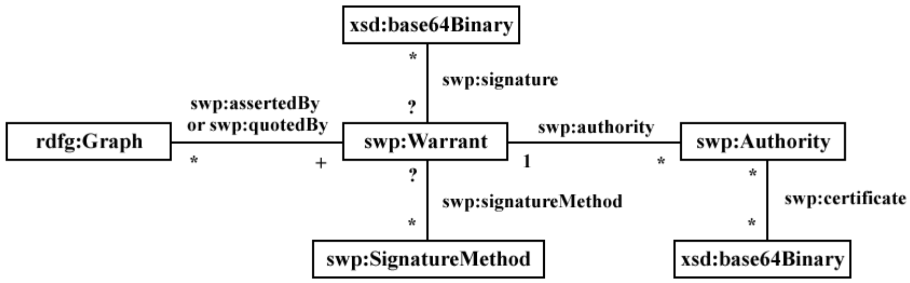

Fig. 1. The Semantic Web Publishing Vocabulary

```
  _:w rdf:type swp:Warrant[6] .
  _:w swp:authority _:a .
  _:a rdf:type swp:Authority .
  _:a foaf:mbox <mailto:chris@bizer.de> }
```

The graph is true when there is a genuine warrant; but so far we have no way to know whether this is in fact the case. A slight modification identifies the graph with the warrant itself:

```
:G1  { :G2 swp:assertedBy :G1 .
       :G1 swp:authority _:a .
       _:a foaf:mbox <mailto:chris@bizer.de> }
```

and the graph describes itself as being a warrant. Suppose further that such a *warrant graph* is in fact authorized by the authority it describes - in this case, by Chris Bizer, the owner of the mailbox: this might be established for example by being published on Chris' website, or by being digitally signed by him, or in some other way, but all that we require here is that it is in fact true. Under these circumstances, the warrant graph has the intuitive force of a first-person statement to the effect "I assert `:G2`" made by Chris.

In natural language, the utterance of such a self-describing act is called a *performative*; that is, an act which is performed by saying that one is doing it. Other examples of performatives include promising, naming and, in some cultures, marrying [2]. The key point about performatives are that while they are descriptions of themselves, they are not only descriptions: rather, the act of uttering the performative is understood to be the act that it describes. Our central proposal for how to express propositional attitudes on the Web is to treat a warrant graph as a record of a performative act, in just this way.[7] With this convention, Chris can assert the graph `:G2` by authorizing the warrant graph shown above, for by doing so he creates a warrant: the warrant graph becomes the (self-describing) warrant of the assertion of `:G2` by Chris. In order for others to

[6] The type triples are implied by domain and range constraints and can be omitted.
[7] The Bank of England uses this technique, by having each twenty pound note bear the text: "I promise to pay the bearer on demand the sum of twenty pounds."

detect and confirm the truth of this warrant requires some way to check or confirm the relationship of authorization, of course: but the qualification of the warrant graph as a warrant depends only on the relationship holding.

A graph describing a warrant is not required to be self-describing in order to be true (it may be true by virtue of some other warrant) and a warrant graph may not in fact be a performative warrant (if it is not authorized by the authority it claims). In the latter case the graph must be false, so self-describing warrant graphs whose authorization cannot be checked should be treated with caution. The warrant graph may itself be the graph asserted. Any Named Graph which has a warrant graph as a subgraph and is appropriately authorized satisfies the conditions for being a performative warrant of itself. For example:

```
:G2 { :Monica ex:name "Monica Murphy" .
      :G2 swp:assertedBy :G2 .
      :G2 swp:authority _:a .
      _:a foaf:mbox <mailto:patrick.stickler@nokia.com> . %%@
}
```

when authorized by Patrick Stickler, becomes a performative warrant for its own assertion, as well as being warranted by the earlier example. As this example indicates, a Named Graph may have a number of independent warrants.

4.3 Publishing with Signatures

Information providers may decide to digitally sign graphs, when they wish to allow information consumers to have greater confidence in the information published. For instance, if Patrick has an X.509 certificate [22], he can sign two graphs in this way:

```
:G1 { :Monica ex:name "Monica Murphy" .
      :G1  swp:assertedBy _:w1 .
      _:w1 swp:authority _:a .
      _:a  foaf:mbox <mailto:chris@bizer.de> }
:G2 { :G1 swp:quotedBy _:w2 .
      _:w2 swp:signatureMethod %%@
swp:std-method-A^^xsd:anyURI .
      _:w2 swp:signature "..."^^xsd:base64Binary .
      _:w2 swp:authority _:s .
      _:s  swp:certificate "..."^^xsd:base64Binary .
      _:s  foaf:mbox <mailto:patrick.stickler@nokia.com> .
      :G2  swp:assertedBy :G2 .
      :G2  swp:signatureMethod %%@
swp:std-method-A^^xsd:anyURI .
      :G2  swp:authority _:s .
      :G2  swp:signature "..."^^xsd:base64Binary }
```

Note that `:G2` is a warrant graph. The `swp:signature` gives a binary signature of the graph related to the warrant. Some method of forming the signature has to be agreed. This is indicated by the value of the `swp:signatureMethod` property on the warrant. We require it to be a literal URI, which can be dereferenced on the Web to retrieve a document. The document describes the method of forming the signature in detail.

Such a method could specify, for example, a variation of the graph canonicalization algorithms provided in [12][8], and choosing one of the XML canonicalization methods and one of the signature methods supported by XML Signatures [17]. Rather than make a set of decisions about these methods, we permit the warrant to indicate the methods used by including the URL of a document that contains those decisions. The URL used by the publisher needs to be understood by the information consumer, so only a few well-known variations should be used.

The publisher may choose to sign graphs to ensure that the maximum number of Semantic Web agents believe them and act on the publication. Using signatures does not modify the theoretical semantics of assertion, which is boolean; but it will modify the operational semantics, in that without signatures, any assertions made, will only be acted on by the more trusting Semantic Web information consumers, who do not need verifiable information concerning who is making them.

The formal semantics of the Semantic Web publishing vocabulary are described in more detail in [13].

4.4 The Information Consumer

The information consumer needs to decide which graphs to accept. This decision may depend on information concerning who said what, and whether it is possible to verify such information. It may also depend on the content of what has been said. We consider the use case in which an information consumer has read a set of Named Graphs off the Web. In terms of the semantics of Named Graphs (section 2.2), the information consumer needs to determine the set A. Information about the graphs may be embedded within the set of Named Graphs, hence most plausible trust policies require that we are able to provisionally understand the Named Graphs in order to determine, from their content, whether or not we wish to accept them. This is similar to reading a book, and believing it either because it says things you already believe, or because the author is someone you believe to be an authority: either of these steps require reading at least some of the book.

The trust policy an information consumer chooses for determining his set of accepted graphs depends on the application area, his subjective preferences and past experiences and the trust relevant information available. Trust policies can be based on the following types of information [10]:

First-hand information published by the actual information provider together with a graph, e.g. information about the intended assertional status of the graph or about the role of the information provider in the application domain. Example policies using the information provider's role are: "Prefer product descriptions published by the manufacturer over descriptions published by a vendor" or "Distrust everything a vendor says about its competitor."

Information published by third parties about the graph (e.g. further assertions) or about the information provider (e.g. ratings about his trustworthiness within a specific application domain). Most trust architectures proposed for the Semantic Web

[8] It is necessary to exclude the last `swp:signature` triple, from the graph before signing it: this step needs to be included in the method.

fall into this category [1, 6, 18]. These approaches assume explicit and domain-specific trust ratings. Providing such ratings and keeping them up-to-date puts an unrealistically heavy burden on information consumers.

The content of a graph together with rules, axioms and related content from graphs published by other information providers. Example policies following this approach are "Believe information which has been stated by at least 5 independent sources." or "Distrust product prices that are more than 50% below the average price."

Information created in the information gathering process like the retrieval date and the retrieval URL of a graph or the information whether a warrant attached to a graph is verifiable or not.

Example trust policies and an example algorithm for choosing which graphs to accept are found in [13]. Further example policies are found in [8, 10].

5 Conclusions

Having a clearly defined abstract syntax and formal semantics Named Graphs provide greater precision and potential interoperablity than the variety of *ad hoc* RDF extensions currently used. Combined with specific further vocabulary, this will be beneficial in a wide range of application areas and will allow the usage of a common software infrastructure spanning these areas.

The ability of self-reference combined with the Semantic Web Publishing vocabulary addresses the problem of differentiating asserted and non-asserted forms of RDF and allows information providers to express different degrees of commitment towards published information.

Linking information to authorities and optionally assuring these links with digital signatures gives information consumers the basis for using a wirde range of different task-specific trust-policies.

Further related work can be found at the TriX and Named Graphs web-site `http://www.w3.org/2004/03/trix/`.

References

1. R. Agrawal, P. Domingos, and M. Richardson. Trust Management for the Semantic Web. In *Proceedings of the 2nd ISWC*, 2003.
2. J. L. Austin. *How to do things with words*. Harvard University Press, 1962.
3. D. Beckett. Redland Notes - Contexts. `http://www.redland.opensource.ac.uk/notes/contexts.html`, 2003.
4. D. Beckett. RDF/XML Syntax Specification (Revised). `http://www.w3.org/TR/rdf-syntax-grammar/`, 2004.
5. D. Beckett. Turtle - Terse RDF Triple Language. `http://www.ilrt.bris.ac.uk/discovery/2004/01/turtle/`, 2004.
6. C. Bizer. Semantic Web Trust and Security Resource Guide. `http://www.wiwiss.fu-berlin.de/suhl/bizer/SWTSGuide`, 2004.
7. C. Bizer. TriQL - A Query Language for Named Graphs. `http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQL/`, 2004.

8. C. Bizer. TriQL.P - A Query Language for Querying Named Graphs Published by Untrustworthy Sources. `http://www.wiwiss.fu-berlin.de/suhl/bizer/triqlp/`, 2004.
9. C. Bizer and R. Cyganiak. NG4J - Named Graphs API for Jena. `http://www.wiwiss.fu-berlin.de/suhl/bizer/NG4J/`, 2004.
10. C. Bizer and R. Oldakowski. Using Context- and Content-Based Trust Policies on the Semantic Web. In *13th World Wide Web Conference, WWWW2004 (Poster)*, 2004.
11. D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0. `http://www.w3.org/TR/rdf-schema/`, 2004.
12. J. J. Carroll. Signing RDF Graphs. In *2nd ISWC*, volume 2870 of *LNCS*. Springer, 2003.
13. J. J. Carroll, C. Bizer, P. Hayes, and P. Strickler. Named Graphs, Provenance and Trust, 2004. Hewlett Packard Labs, Technical Report HPL-2004-57.
14. J. J. Carroll and P. Stickler. RDF Triples in XML. In *Extreme Markup Languages 2004*, 2004.
15. Creative Commons Website. `http://creativecommons.org/`, 2003.
16. E. Dumbill. Tracking Provenance of RDF Data. Technical report, ISO/IEC, 2003.
17. D. Eastlake, J. Reagle, and D. Solo. XML-Signature Syntax and Processing, RFC 3275. `http://www.w3.org/TR/xmldsig-core/`, 2002.
18. J. Golbeck, B. Parsia, and J. Hendler. Trust Networks on the Semantic Web. In *In Proceedings of the 7th International Workshop on Cooperative Intelligent Agents, CIA2003*, 2003.
19. P. Hayes. RDF Semantics. `http://www.w3.org/TR/rdf-mt/`, 2004.
20. A. Ibrahim. Agent Communication Languages (ACL). `http://www.engr.uconn.edu/~ibrahim/publications/acl.htm`, 2000.
21. Intellidimension. RDF Gateway - Database Fundementals. `http://www.intellidimension.com/pages/rdfgateway/dev-guide/db/db.rsp`, 2003.
22. ITU-T. Information Technology - Open Systems Interconnection - The Directory Authentication Framework. X.509, 1997.
23. G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. `http://www.w3.org/TR/rdf-concepts/`, 2004.
24. R. MacGregor and I.-Y. Ko. Representing Contextualized Data using Semantic Web Tools. In *Practical and Scalable Semantic Systems (workshop at 2nd ISWC)*, 2003.
25. M. Marchiori. The platform for privacy preferences. `http://www.w3.org/TR/P3P/`, 2002.
26. A. Seaborne. RDQL - A Query Language for RDF. `http://www.w3.org/Submission/2004/SUBM-RDQL-20040109`, 2004.
27. P. Stickler. RDFQ. `http://sw.nokia.com/rdfq/RDFQ.html`, 2004.