Draft URI Strategy for the NL Public Sector

For feedback by the expert community present at ODW 2013 in London

Introduction

This document focuses on the technical considerations, made during the process of specifying a national strategy for minting URI's to identify information of the Dutch government on the internet. It is a draft only, no definite decisions or recommendations are made and all statements made here can be subject to change.

In 2012 the EU Directive INSPIRE gave momentum to the development of such a national strategy. The directive [1] states that:
"The implementing rules shall address a common framework for the unique identification of spatial objects, to which identifiers under national systems can be mapped"
with the recommendation to connect this framework for spatial objects to a general national framework.

We were inspired by the UK recommendation of data.gov.uk, described in
"Designing URI Sets for the UK Public Sector." [2]

In january 2013 ISA published a comprehensive document with useful recommendations: "D7.1.3 - Study on persistent URIs, with identification of best practices and recommendations on the topic for the MSs and the EC"[3]

Making use of these guidelines where possible, a group of stakeholders and experts deducted a strategy for the Netherlands. Involved parties were, among others: Geonovum, Knowledge Center for Official Government Publications (KOOP) and the Tax Service. They formulated a strategy of which this document covers the technical parts, in order to get feedback from an expert community at the ODW 2013.[4]

The quest

During the study of the starting documents we learned some things we didn't find in the documents. Some of them useful, some of them less promising.

Functions of Linked Data

The whole exercise of defining a URI strategy is setup because we realize it is useful to re-use already defined concepts. Abstract concepts, defined in standard models, and real life things that are important because we keep a lot of information about them. And to make combinations of different information collections we give those things a name or number, so that we can all use the same identifier to point to the same object. For ages we keep registers for these things.
Now when we look at the Semantic Web, this is no different. We want the data in different business applications to refer to the same things in the same registers, using the same abstract model, so that we can merge data sets from different applications.

Functions of Linked Data

A. Standard (ex: SKOS, DC, FOAF, ECLI, OWMS)

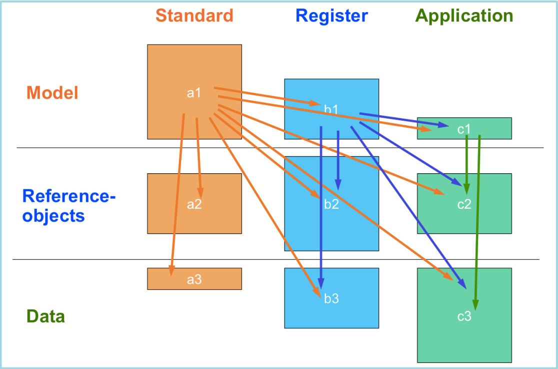
- Semantic model (ontology) for a domain or sector
- Vocabulary for the model

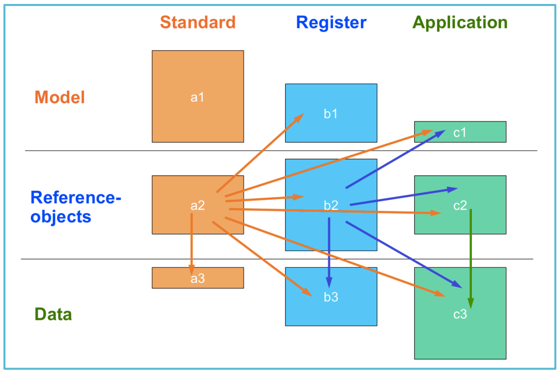
B. Register (ex: roads, schools)

- Administration of ‘Things’
- Mints URI’s for things, to be re-used
- Uses vocabulary of the standards

C. Application (ex: weather, trafic, population)

- Provides data about the things in the registers
- Uses URI’s from register and vocabulary of the standards





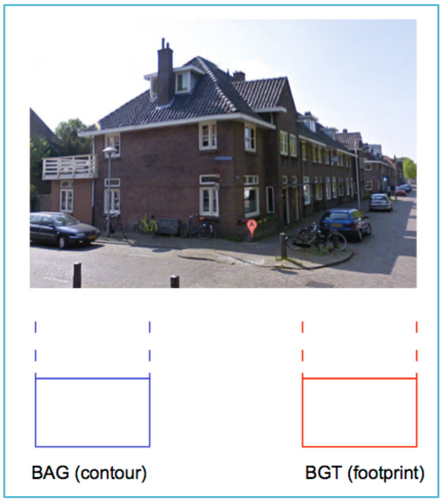
No register? No identifier!

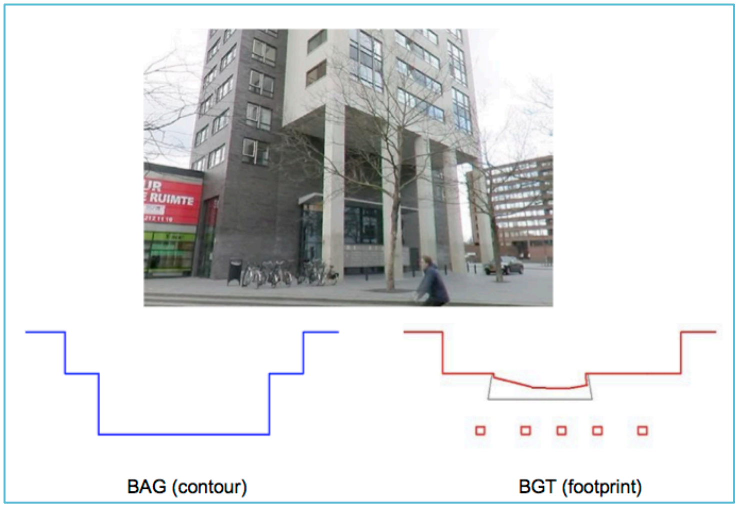
When thinking about choosing identifiers for real life things, we realized that things in real life never have a suitable identifier by themselves. As soon as we try to mint identifiers for real life things we need to register them. Identifiers only exist in registers!

For things that need to be referenceable often there are registers already. We don’t need new ‘Linked Data Identifiers’ to identify things. We can simply construct URI’s from the records in existing registers. That has some implications: If we keep different registers for different purposes, we might mint different URI’s, which in some cases seem to refer to ‘the same’ object in real life. Is this a problem? No!

But since we have those different registers, it is apparently worth the cost to keep different registers, hence they have a different goal and therefor it might be useful to have more than one URI pointing to what seems to be “the same thing”. In almost all cases you will see there are differences in the models of the two registers, which means that the context is different. In most cases there are examples where no exact one-to-one mapping is possible.

An example: in NL there is a register for Addresses and Buildings (BAG) and there is a register for Large-Scale Topography (BGT). Both registers contain houses and offices. But a building in the BAG refers to the construction as a whole, whereas a building in the BGT refers to its ground level footprint. So an arc-shaped building corresponds to one object according to the BAG model, but to two objects in the BGT model.





It is no problem if a URI contains a reference to its register. Or even stronger: we recommend making this reference part of the URI-pattern.

Our lesson:
"No register? No identifier!"

Another paradigm?

It took quite some meetings before we got to the point where we all began to understand the implications of using http-URI's to identify things. And still it is sometimes confusing that the URI 'only identifies the thing', but non the less should be dereferenceable or actionable and lead you (303) to a document about the concept that was identified.

W3C recommends[5] http-URI's:

- Use URIs as names for things
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
- Include links to other URIs. so that they can discover more things.

In order to circumvent this confusion we examined the possibility to use URN's to name things and have resolvers that can interpret those URN's and provide information about those things. The recommendation would then be:

Alternate recommendation:

- Use URNs as names for things
- Use REST-services as resolvers for those URNs, so that people can look up those names.
- When someone looks up a URN, provide useful information, using the standards (RDF*, SPARQL)
- Include links to other URNs. so that they can discover more things.

Advantages of this approach:

- People will easier understand the difference between
  - {URN}: the identifier of the thing and
  - http://{resolver}/{URN}: the location of information about the thing
- No 303 redirects needed
- Multiple resolvers possible. E.g:
  - The resolver of the register itself
  - a generic resolver for all LOD URN's
  - a sector resolver for all URN's of registers in a sector
  - every application that uses URN's can make a page with an application specific view on the referred thing.
- URN's are not bound to domains
- No nested http-uri's

Drawbacks of this approach:

- It is another paradigm, which is not in line with the current W3C recommendation
- It needs a new, global URN-scheme to be succesful
- There needs to be an indication of the register in the URN to know which resolvers would possibly be able to resolve the URN.
- We would need a 'register of registers' (ICANN for the GGG)
- With reference to the corresponding resolver

So for us this was too big to handle and we abandoned the idea, but we would still like to share the idea with you.

Conclusion

At this point in time we want to draw a provisional conclusion, formulate a strategy and investigate the consequences of it. Besides some technical implications, there are of course some organizational aspects to setting up the strategy. We have to investigate to what extent we should centralize parts of the solution, or rather keep things federated.

The goal of the URI-strategy is that it is appealing for organizations to re-use standards and registers of others and to make their own models and concepts referenceable for others. So the strategy should support an eco-system, which is trustworthy, intuitive to adapt to, persistent and easy to use.

The pattern that fulfils that strategy is the pattern suggested by [2] and [3] already.

http://{domain}/{type}/{concept}/{reference}

{domain}

http://{domain}/{type}/{concept}/{reference}

The{domain} consists of the internet domain and a path: {domain} = {internet domain}/{path}. We recommend that the domain of a URI contains a reference to the register it comes from, either in the internet domain, or in the path, in case there are more registers kept at the same domain. For smaller registers the latter is sometimes the case. For bigger and more important registers we recommend there is no path at all to keep the URI as short as possible.

In order to make URI recognizable for humans and give user confidence about what a URI is meant to be and how it will behave we consider reserving a generic domain for URI's of the NL Public Sector. For instance: 'data.gov.nl' or 'data.overheid.nl'[1].

The ideal domain would be: {register}.data.gov.nl/ but the following parterns are considered:

- {sector}.data.gov.nl/{register}
- {register}.data.gov.nl/{path}
- and even {yourdomain}/{yourpath}

This raises some questions, which are partially organizational. The owner of the data.gov.nl domain will have to keep an administration of the sub-domains that exist and he has to take care that DNS-routing of requests to servers for each sub-domain is properly managed. The profit is in the fact that the domains will resemble each other, while still scalable because load is balanced using DNS. Every register is responsible for it's own load without a central burden.
Another question: "Does a {path} between the internet domain and the rest of the URI lead to complications?

{type}

http://{domain}/{type}/{concept}/{reference}

The {type} indicates which kind of URI this is.

- 'id': identifier of real life object *in a register*
- 'doc': documentation about the real life object by this register
- 'def': definition of a term in an ontology

For def-URI's we recommend to use hash-URI's. For two reasons:

- recognisability of the distinction between terms from the ontology and terms to identify instances.
- To be able to find the whole ontology in an easy way. You will find the whole model at: http://{domain}/def

UK-strategy recommends slash URI's for ontology terms. Why?
Sometimes it is hard to separate model from content. Some people prefer to give everything an id-URI and make no such a sharp distinction.

{concept}

http://{domain}/{type}/{concept}/{reference}

The concept indicates the kind of thing the URI identifies. Usually it is a class name. A lot of conventions can be made about this part. Conventions increase intuitiveness, but can be hard to learn. Mistakes are easily made. To name a few possible conventions:

- in the 'id'-URI you often see the concept in lowercase, whereas in the 'def'-URI UpperCamelCase is the default for classes and lowerCamelCase for properties;
- you could also use underscores ('_') or hyphens ('-') to separate words;
- use singular terms for concepts
- avoid all other characters than a-z, A-Z, 0-9

How important is each convention?

[1] "Overheid"@nl = "Government"@en

{reference}

http://{domain}/{type}/{concept}/{reference}

The {reference} is usually a string which is used in the existing register already. Some conventions here:

- use names or abbreviations of names if they are available and do not lead to collisions
- avoid all other characters than a-z, A-Z, 0-9, underscore ('_'), hyphen ('-'), brackets ('(' and ')') or slash ('/')
- if you use dates in URI's to point to versions, use the W3CDTF format: yyyy-mm-dd

References:

[1] http://inspire.jrc.ec.europa.eu/
[2] https://www.gov.uk/government/publications/designing-uri-sets-for-the-uk-public-sector
[3] https://joinup.ec.europa.eu/sites/default/files/D7.1.3%20-%20Study%20on%20persistent%20URIs_0.pdf
[4] http://www.w3.org/2013/04/odw/
[5] http://www.w3.org/DesignIssues/LinkedData.html