Levels of Open Data

The levels in the Quality Guide for "Providing data" ("Tillhandahålla data") are linked to the steps identified by the work of E-delegation[1], an expert group appointed by the government. Their steps were drawn, in the main, from Tim Berners-Lee's 5-Star open data rating scheme.

The 5-start ladder is well-represented by the following diagram.

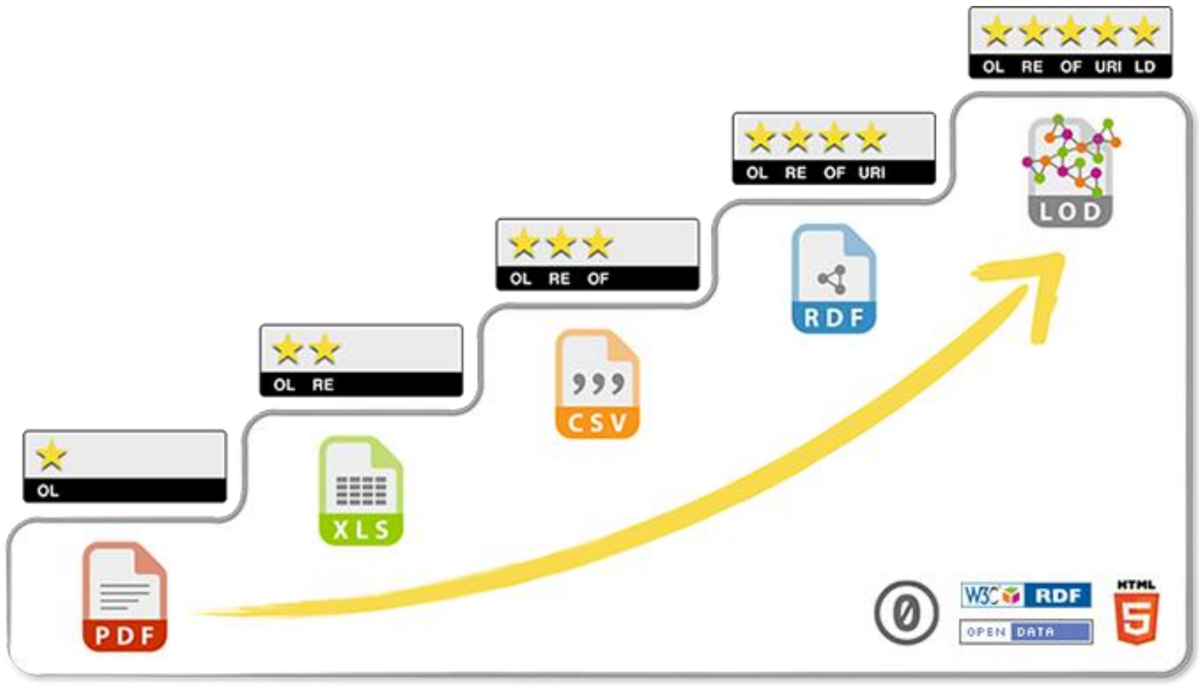


Figure 1: 5-Star steps [From-http://5stardata.info/en/]

Level one

The minimum level for Providing Data is linked explicitly to the first step of the E-delegationen's levels: ***"Steg 1: Publicera informationen på webben i det nuvarande formatet"***
This, in turn is based on Berners-Lee's first star:
"Available on the web (whatever format) *but with an open licence, to be Open Data"*

This is perhaps the single biggest step and culture change for an organisation if it embraces the concept of Open Data. The requirement for data to be open is that it needs to be available online and declared open using an open licence.

A good way to publish online is to register a metadata catalogue – for instance SLU's TILDA portal when it comes online.
Little more need be said about the technical requirements, since the only requirement is that it be placed online with an open licence.

[1] In the section " 4.4.6 På vilket sätt ska tillgängliggörandet ske?" in their report "Vidareutnyttjande av offentlig information"(2014)
Now found here: "https://oppnadata.se/5-stjarnemodellen/"

Open Licenses

Examples of popular and valid open licences are those specified by the Creative Commons with their CC0 (Public domain) and CC-BY (open licence, but with terms of requiring attribution to the data producer and also the requirement to highlight any changes made to the data). SLU open data should, in the main, be published with CC0.

Creative Commons is designed for creative works primarily, though this does not harm their legality or usability. For a more data-specific licence then Open Data Commons licenses PDDL (Open Data Commons Public Domain Dedication and License) and Open Data Commons Attribution License (similar to CC-BY).

Level two

The level explicitly equates to step 2 from E-delegationen: ***"Steg 2: Publicera informationen i maskinläsbar strukturerad form"***
Which, in turn is based on Berners-Lee's second star: "Available as machine-readable structured data"

This again is actually quite simple.
At the lowest level one may have data tables presented in image forms, perhaps PDF files containing scanned pages from old texts, or web pages featuring similarly scanned images. While this would be ok for the lowest level, level two requires files in "machine-readable" files. For instance tables in the older Excel file format (XLS) are machine readable.
As a simple test that works for simple datasets, data is "machine readable" if you can copy it out and paste it in new file.

A secondary factor is to make spatial/geographical data (*geodata*) available using the most modern Swedish standard National Map projection SWEREF 99 TM.
Since there are many tools and function libraries that enable conversion of spatial data from one map projection to another.
GIS tools like ArcGIS and QGis can perform such conversions fairly simply.

Level Three

Reflects step three: *“**Steg 3: Publicera informationen i ett öppet format eller i en öppen standard**”*
This in turn is based on Berners-Lee’s third star: “make it available in a non-proprietary open format”

This step is, in some senses more complex and controversial. The idea being simply to make data more usable by more people. Therefore it is better if it is not limited to a single application for use, especially costly programs like Microsoft Excel or Esri’s ARCGis.

However companies like Microsoft, ESRI and Adobe are often taking the steps to publish open standards from their applications. For instance using the newer Office Open XML formats (e.g. XLSX, DOCX rather than the older xls and doc).

It can then be argued that such files, as well as ArcGIS files in Shapefile formats, are now published in an open standard.
It can be also argued that, especially Microsoft’s products, are not sufficiently open, nor a sensible choice. It’s worth mentioning that the E-delegation considered them sufficiently open.

At the most basic level data can be presented in pure text format – typical web-based examples being standard web pages (HTML), or data structures like XML and JSON but a good example for those using more file-based data being CSV, and in many cases having this along with an Excel version of a datatable would provide users with good alternatives for use

If however the data-type or size or complexity makes such pure-text formats inappropriate – for example large databases may not be reasonably “exportable” - there are often open (or semi-open) options worth using.

Level Four

Reflects E-delegationen’s fourth step: ***”Steg 4: Gör informationen åtkomlig via ett API”***

Can be considered as related to Berners-Lee’s fourth star: “use URIs to denote things, so that people can point at your stuff”

It is worth noting here that the E-delegationen step does not match Berners-Lee’s fourth star. There is a considerable amount of overlap between these two declarations and a little depends on what one understands an API (see below) to include.

In terms of our data quality guide you need to be using URIs (see section below) to identify data elements in order to reach level four. This will most normally be done via an API.

The central idea of this level is that data needs to be identifiable - and that users need to be able to check the data source, whether updates have occurred and so on. A simple API that delivers data does not necessarily do either. Which is why the E-delegationen’s guide suggests using methods to identify changes in data (giving Atom Syndication format as an example) and using unique identifiers to access data (which are usually URIs).

The question of API construction is very much based on the size, complexity and atomic value of data. An API suggests that a user may want to get at a specific part of the dataset and that will be valuable to them.

If the data being provided is a simple table then it may well be enough to provide it with appropriate URIs for identification. Providing that the data does not then change this will provide the adequate functionality for level four. It’s worth noting that an example page given showing “4-star data” according to the Berners-Lee scale is not an API, but a simple web page.

What is an API?

The acronym API stands for Application Programming Interface which perhaps doesn't make it a lot clearer.
An API is a programming concept, the definition of an interface which provides access to functions and values from one programming application or system out to others. Windows API was a way to make Windows functions available to other applications which runs on the Windows operating system. This allows windows applications to look familiar, using standard design elements.

When we are speaking about API now we are usually speaking of Web API. This means, in essence, that functions and data are made available to other users via the web.

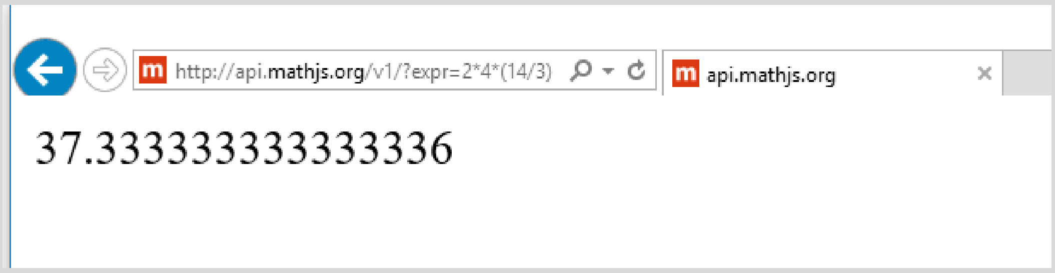
Many API use the standard web HTTP communications protocol. There are four standard "verbs" associated with there: GET, POST, PUT and DELETE. These allow programmers to interact with fairly complex and powerful systems on a website

The most straightforward way to get a feel for it is to look at a GET command. Unlike the others it can be issued by using the address bar in your browser.

There is a simple mathematics API at mathjs.org.
The API's URL is http://api.mathjs.org/v1/

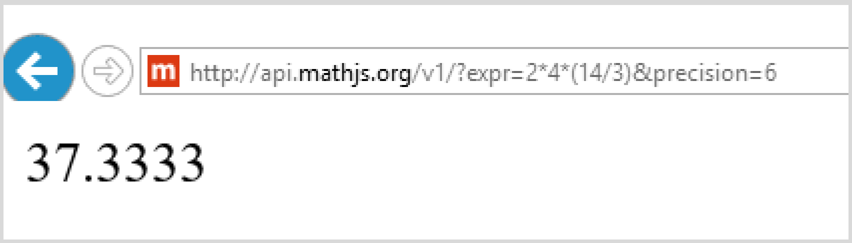
We can then pass it some data to work with. The simplest way is to add one query parameter which according to the documentation is simply "expr".
If we want to calculate an expression like 2 * 4 * (14/3) then we add this to the URL and put it in the address bar: http://api.mathjs.org/v1/?expr=2*4*(14/3)
Everything after the "?" is a query string.

Entering this and pressing return sends a 'GET' instruction to the server and it returns the result of the calculation as simple text.



```
37.333333333333336
```

If we wanted to add a second parameter we can give it the "precision" parameter using an "&" and the appropriate value: http://api.mathjs.org/v1/?expr=2*4*(14/3)&precision=6



```
37.3333
```

Of course, it is more common that we want to use an API to fetch data (rather than calculate things). But it is the same method used: a GET command with parameters.

StackExchange (a handy collection of sites for asking and answering questions, frequently technical ones) has an API which provides the following function: https://api.stackexchange.com/2.2/users/

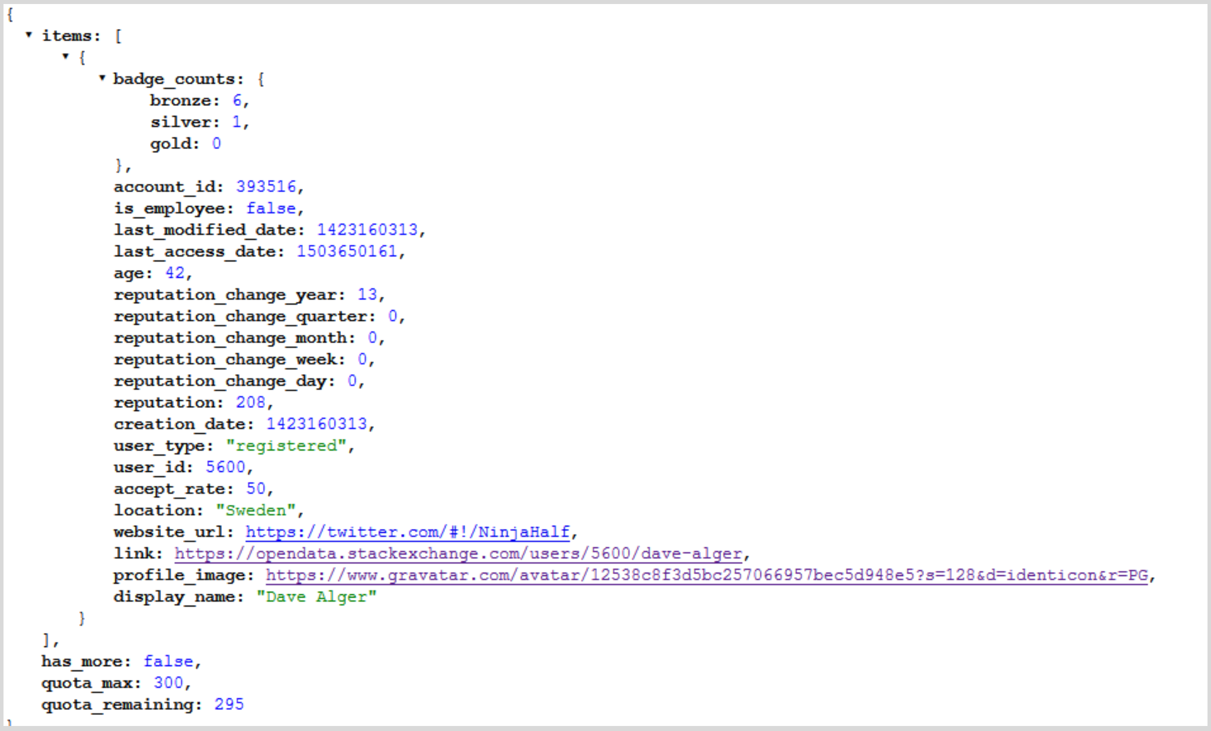
If we add a parameter "?site=opendata" we get some user values.
https://api.stackexchange.com/2.2/users/?site=opendata

Which just returns text (your browser may want you to open this as a download, you can open it in notepad if you want to)– this is actually in JSON format which is a very frequently used data format for Web APIs (which of course is already at Level Three). While the e-delegation recommendation is to use XML one can also use JSON or plain text depending on the data being returned. Many experts now recommend JSON as the best data form.

According to the API's documentation one can add a user id to the URL parameters. In this case it is not added after the "?", but is part of the URL path:

https://api.stackexchange.com/2.2/users/5600?site=opendata

The result, which is my user data, is shown below with some simple formatting to make it readable.



```
{
  ▾ items: [
      ▾ {
          ▾ badge_counts: {
                bronze: 6,
                silver: 1,
                gold: 0
            },
            account_id: 393516,
            is_employee: false,
            last_modified_date: 1423160313,
            last_access_date: 1503650161,
            age: 42,
            reputation_change_year: 13,
            reputation_change_quarter: 0,
            reputation_change_month: 0,
            reputation_change_week: 0,
            reputation_change_day: 0,
            reputation: 208,
            creation_date: 1423160313,
            user_type: "registered",
            user_id: 5600,
            accept_rate: 50,
            location: "Sweden",
            website_url: https://twitter.com/#!/NinjaHalf,
            link: https://opendata.stackexchange.com/users/5600/dave-alger,
            profile_image: https://www.gravatar.com/avatar/12538c8f3d5bc257066957bec5d948e5?s=128&d=identicon&r=PG,
            display_name: "Dave Alger"
        }
    ],
    has_more: false,
    quota_max: 300,
    quota_remaining: 295
}
```

This perhaps isn't so very useful, but one can add an extra parameter into the path "questions" and this will provide a list of the questions I've asked on the site:

https://api.stackexchange.com/2.2/users/5600/questions?site=opendata

This site gives access to data in an open way, it provides it in pure text format via an API. But it is not enough to make this Level Four because it's not using URIs to identify all or many of its things.

What is a URI?

URI stands for Uniform Resource Identifier – it's a string of characters used to identify a "resource". The most common form of URI is a URL (Uniform Resource Locator), i.e. a web address, and this is probably a convenient way of thinking about a URI.

A URL features a 'protocol' – such as HTTP:// or FTP://
A URI doesn't necessarily need to feature that type of information.

As an example we can take a URL that identifies a dataset: http://d-nb.info/gnd/4151140-2 (it's a data record for the German tree species Eichen i.e. Oak).

If you follow this link it will use the web portal of the Deutsche National Bibliothek to fetch a record and the URL comes as:
https://portal.dnb.de/opac.htm?method=simpleSearch&cqlMode=true&query=idn%3D041511409

As one can see there's a lot more information in the second URL and how it accesses data from the portal – and while it points correctly to the record at the moment, it's not a real identifier. The actual ID (the first URL given) is the one that actually identifies the record and should always point us to the dataset.

The identity URI also doesn't contain a query string (i.e. bits after the "?"). While these are very common in producing APIs they are, according to many knowledgeable open-data experts, not valid as part of an ID. So a URI should be something like: *slu.se\dataset\record\1* rather than *www.slu.se\dataset\recordfetcher?table=thing&id=1*

The value of being able to uniquely identify a data post is that it is then possible to link data sets. Which brings us to the final level.

Level Five

E-delegations fifth step: ***"Steg 5: Publicera informationen som länkad data"*** maps to the fifth star, "link your data to other data to provide context"

At the *fourth level* we have made it possible for others to link to our dataset and contribute to the linked data cloud. But if we link our data to other sources we add value to it by providing more information and context.

There are many maps generated of the Linked Data cloud, below is an old one from 2010, there are many more nodes now and it's hard to get an overview.

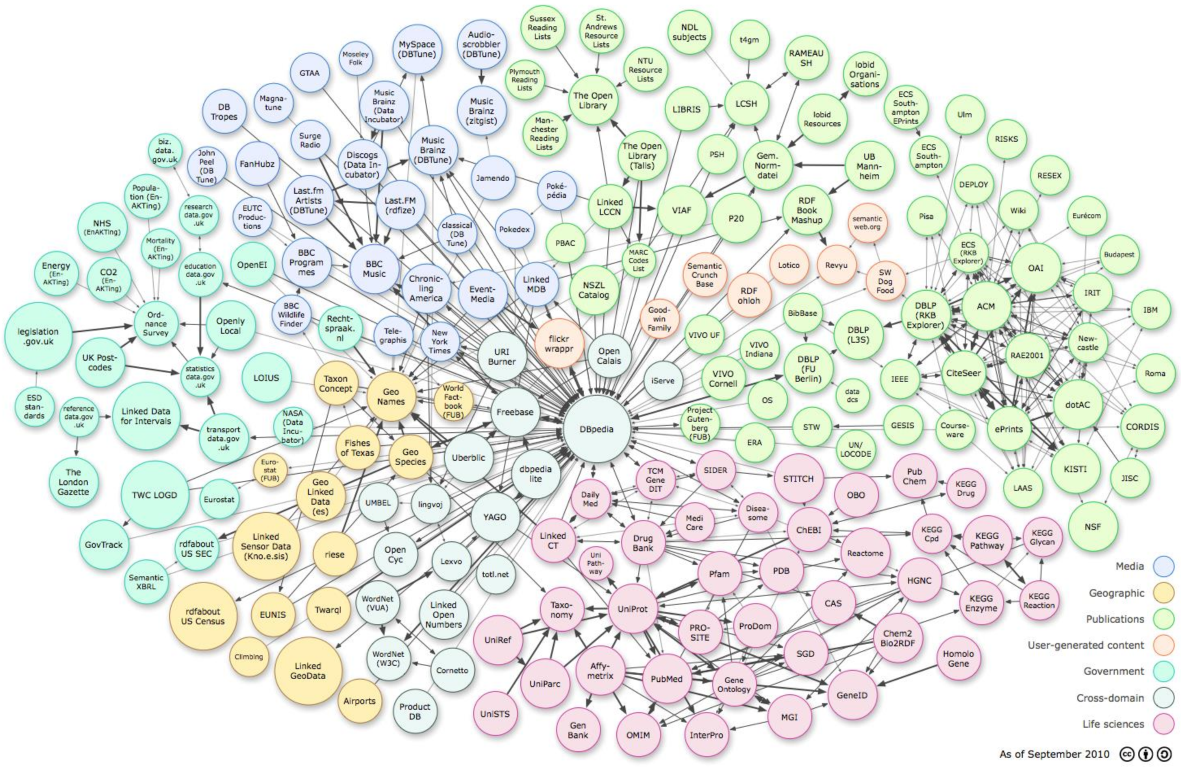


Figure 2 A diagram of the Linked Data cloud[2]

One of the largest circles near the middle is Dbpedia. This site presents data from Wikipedia in linked data form. It is therefore a great resource that people use to define things. But Dbpedia also points to other datasets. If we look at their page on Oak (http://dbpedia.org/page/Oak) we can see a lot of information about the species. But if one looks down at the section with a label "owl:sameAs" we can see that one of the links there is back to the Deutsche National Bibliothek page.

The advantage here is that it is possible to link from Dbpedia's data to a data record referencing Eichen in a German context, and to a French source describing Chêne data.

It is also possible to add links describe our data in other ways. The description owl:sameAs is part of the Web Ontology Language (OWL), so that this relationship is understood. Other "vocabularies" exist, like the RDF Schema, Dublin Core metaterms and others. Using these vocabularies we can

[2] Linking Open Data cloud diagram 2017, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/

specify our units of measurement in terms that allow people of any language to understand. We can use simple references to authors, organisations and so on. We can also link to wider concepts to allow people to understand our data without needing additional documentation.

Kvalitetsguiden also encourages data producers who cannot connect to existing definitions and descriptions online begin to create linkable documentation to strengthen the dataset's description and usability.

If you're interested in the Linked Open Vocabularies that exist then the Open Knowledge Foundation has an interesting site which shows many: http://lov.okfn.org/dataset/lov

What is RDF?

Linked data is expressed in Resource Description Framework (RDF) form. RDF/XML is a popular format, but it is also possible to use RDFa which is Linked Data in HTML format, or JSON-LD.

Most examples shown by Linked Data have shown the popular RDF/XML format, and is the format recommended by E-delegationen. (see below)

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/">
 <foaf:Person>
   <foaf:name>John Lennon</foaf:name>
   <foaf:gender>Male</foaf:gender>
   <foaf:birthday>1940-10-09</foaf:birthday>
   <rel:spouseof>Cynthia Lennon</rel:spouseof>
 </foaf:Person>
</rdf:RDF>
```

One can also format data in a normal HTML document using tags in the format of RDFa.

```
  <div vocab="http://schema.org/" typeof="Person">
     <img
src="https://en.wikipedia.org/wiki/John_Lennon#/media/File:John_Lennon_1964_001_cropp
ed.png" property="image" alt="Photo of John Lennon"/> <br>
     <b><span property="name">John Lennon</span></b> was married to
     <div property="spouse" typeof="Person">
          <span property="name">Cynthia Lennon</span>
     </div> and was born <span property="birthDate">1940-10-09</span>.
   </div>
```

These tags do not alter the visual representation of the data – however the RDFa description of the data allows the data to be structured. Google are providing tools that allow you to check whether data is well-formed. For instance the above example can be analysed in Google's Structured-Data Testing Tool

Both these methods require substantial modification to existing datasets and documents – but they do not make the data any more difficult to consume or understand from a user point-of-view.

A new and popular format of RDF is JSON-LD. It is built on the structure for JSON data which is fairly clear and doesn't require too much additional data or tags to format – and it can be fairly simply read and is a very popular data format for use in web-services. (See an example below)

```
{
  “name”: “John Lennon”,
  “born”:”1940-10-09”,
  “spouse”:”Cynthia Lennon”
}
```

In order to make this data 'linked data' it needs to be connected to vocabularies and can also be connected to other datasets. For instance:

```
{
  "@context": "http://json-ld.org/contexts/person.jsonld",
  "@id": "http://dbpedia.org/resource/John_Lennon",
  "name": "John Lennon",
  "born": "1940-10-09",
  "spouse": "http://dbpedia.org/resource/Cynthia_Lennon"
}
```

In this sort of example, with this sort of mapping it's often possible to raise level four data to level five by providing these contextual clues.
Now it doesn't matter which language has been used since the connections to existing vocabularies (person.jsonld here, but schema.org used above would also be appropriate) enable links to other languages and data sets.