Simple five level Open Data API evaluation model

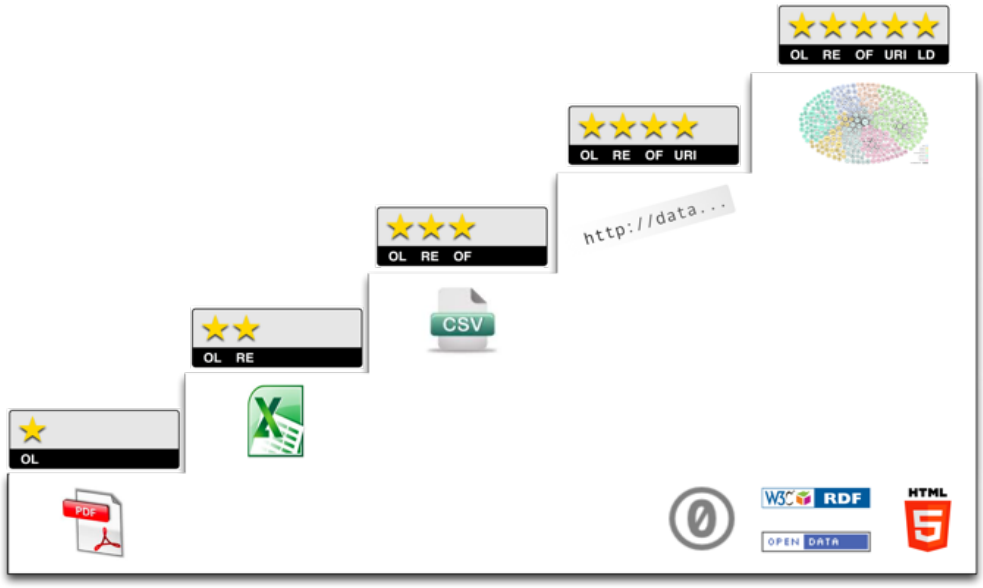
Submitted on 06 Nov 2013 by Jarkko Moilanen

Whatever the selected API management is, we still need tools to evaluate APIs. APIs come in different flavours; some good, some not too good and a wide variety in between. We need tools to evaluate APIs or to be more precise - we need to evaluate API providers and communities.

Evaluation frameworks give us tools to quickly give a rating for API and to present the ratings for developers. Developers (aka API key owners) should be able to vote on APIs. Developers should also be able to submit bug reports regarding any API. Building a central or distributed API management solution discussed in previous post, we could also gather bug and feature dialog in the same platforms. This would make APIs related dialogue more visible and accessible. It would also remove the need for separate dialogue systems and thus lower the costs. Now, let's get back to the evaluation model suggestion.

Follow the logic of the deployment scheme for Open Data

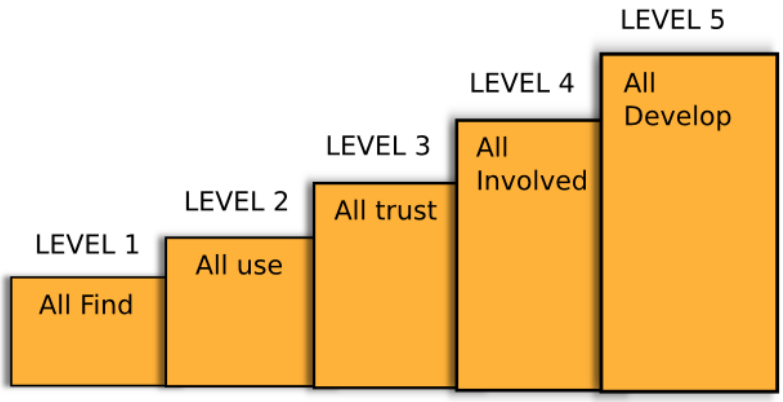
I've done an initial 5 step model to "give stars" for APIs. The model follows the familiar logic in Tim Berners-Lee's 5-star deployment scheme for Open Data below.



http://5stardata.info/Five step Open Data API evaluation model

Five step Open Data API evaluation model

Each main plane has child items, and each of them can get one point. At present, the maximum score is asymmetric, i.e. a maximum of 23 points.



Level 1: All find (4)

- From a single source (the portal), are all of the information or links to them available?
- Is updated documentation available?
- Are there examples of API requests seen as part of the documentation?
- Are there examples of the API request returned data?

Level 2: All use (5)

- Does it support the use of JSON and/or XML?
- Are data license details given through the API?
- Are Terms of Use clear and easily accessible?
- Does returning data include metadata?
- Does authentication exist? Oauth (2), or other?

Level 3: All trust (5)

- Is Analytics API public and real/up to date?
- Is Error Handling in place and documented?
- Does it support queries and use of cache?
- Is the background a big ripe entity or a company that is engaged in the development and maintenance of the API?
- Is it available for all (business, communities, individual developers)?

Level 4: All involved (5)

- Are API SDK's for one or more of the environment are available?
- Are there examples of code in one or more of the programming language?
- Is there a growing community (and the location in the network) to consult if needed?
- Is there an API Playground for testing and getting familiar with?
- Is documentation linked to code examples and back again?

Level 5: All develop (4)

- Is code visible/can be cloned?
- Can Bugs be reported in a public place and is dialogue public?
- Is the API's license known, and does it allow further development and re-use?
- Is the API's development roadmap known and is it visible for all (how to develop and at what stage)?

Evaluation data crowd-sourced

Data which evaluates listed APIs would be crowd-sourced. For example, the developers who request API keys could be a suitable audience to audit APIs based on the described 5 steps.

To recap the points model: Each step has 4-5 point items. Each item is a claim.

- If a claim regarding API at hand is true, it is recorded as green (1 point).
- If it's not known whether a claim applies to API, it is recorded as turquoise (0 points).
- If a claim does not apply to selected API, it is recorded as red. (0 points) and lastly
- If there is no data regarding the claim, it is recorded as gray (0 points).

After a while, when developers have evaluated APIs, data could be automatically transformed into easy to use graphs or tables such as the below illustration. The colours of each bar could express the most selected option; the more developers evaluate one item as "green" than "gray", the bar is green. That way system could use "developer votes" instead of relying on single time data entry.

Of course the model needs to be clarified, but it should at the current form provide a starting point for building scalable crowd-sourced API evaluation systems. This kind of tool would help in identifying shortcomings of each API (what to develop next) and it would also help developers in choosing which API to use.