.SE (The Internet Infrastructure Foundation)
Mats Dufberg <mats.dufberg@iis.se>
Patrik Hildingsson <patrik.hildingsson@iis.se>
2014-12-19

Comments on the LGR tool requirements

Comments on the "LGR Tool Set Specifications" <https://www.icann.org/public-comments/idn-lgr-2014-12-03-en> and <https://www.icann.org/en/system/files/files/idn-lgr-toolset-reqs-03dec14-en.pdf> sent out by ICANN.

1 Summary

Our conclusion is that ICANN seems to be willing to invest considerable amount of time and resources (money) into a tool to be used by the VIP program where the use is not time critical instead of focusing on the broad use of IDN tables where the use is time critical and the use of standardized IDN tables will have a broad impact. We also question that it is correctly spent resources to create a complex web interface for creating LGRs.

2 Background

First some background as we see it. There are two major use cases for LGR:

1. Managing IDN TLDs in the root zone.
2. Managing IDN SLDs in TLDs.

If LGR is successfully implemented as the standard for IDN tables, then the second use case will be the high volume use case. The promise of LGR is to make it possible to have an IDN table for a TLD that can be used by all stakeholders to verify if an IDN label is compatible with the table:

1. The registry
2. The registrar
3. The registrants
4. ICANN
5. The Internet community

With the LGR IDN table all parties can verify if a certain string (domain) matches the table or not. The crucial requirement for this to be possible is that there is a tool kit that can be used for that, and that the toolkit can be integrated in the backend system (EPP) at the registry, the EPP client at the registrar and in stand-alone tools.

When further splitting the TLD use case up, we can see a number of sub-use cases, in the order of expected frequency:

1. Validating a proposed string (domain) against an LGR IDN table.
2. Visually inspecting an IDN table.
3. Comparing two IDN tables, especially two IDN tables that have the same label, e.g. the same language.
4. Creating an LGR IDN table.

Of those sub-use cases validating a string, will be the most time critical if LGR is fully implemented at the TLDs. I.e. the time it takes to verify a string against a table must be very short, almost instantly.

Creating an LGR by hand is time consuming, but it is not time critical. The creation of IDN tables will be done once by e.g. the TLD registry. It will maybe be updated, but usually it will be used for a long time once it is created. In many cases the registries will not even create the table, they will copy the table for some other TLD.

For the Generation and Integration Panels of the VIP program the situation is a little bit different. They must create the tables, because they have none to copy from, but still the work is not time critical.

3 The Specification

Now we go back and compare our view with the suggested Specification published by ICANN (see links above). The focus of the Specifications is for the use of LGR for the Generation and Integration panels in the VIP program. It is not surprising since LGR was originally created for that purpose. In that work the creation of IDN tables is a major task, but if we want to ensure that LGR is spread in the TLD world, then the focus should be twisted.

1. First create an engine that reads a LGR table and validates labels against the LGR table. That engine must be quick and reliable.
2. In the second step create report module that can present an LGR table in a human readable format.
3. In the third step create an import module that can read an IDN table according to RFC 4290 or RFC 3743 and create a LGR table. Since those formats are more limited than the LGR format, there are details such as contextual rules that must be added by hand.
4. In the forth step create a web based tool for creation and updating of LGR tables to the extent that is motivated. Complex tables must be updated by hand.

3.1 Web interface for programming

Creating an LGR can be seen as a programming task. The idea behind the Specification is that there should be an interface for creating an LGR table without having to handle XML, i.e. a programming interface for non-progammers. After creating an LGR, regression tests will be necessary, i.e. testing a number of legal and illegal strings

against the LGR. If any of the tests fails analysis of the LGR must be done. In non-trivial cases such an analysis require programming skills.
Instead of asking programmers to create a complex interface for creating LGRs, it is probably better to use the resources to let persons with programming skills help with the creation of the LGRs in the first place.

3.2 Complex requirements

In the Specification we see requirements that will be very complicated to implement and where it is hard to see useful use cases. E.g. that the tool should be able to create intersection and union, respectively, of two LGRs. We would like to see a better balance between cost, usefulness and complexity. Requirements that are complex to implement not only increase cost, but it also increase the risk of errors.
The Specification points out the TLD use of LGR as one of the use cases that the specification is targeting. A tool or rather a toolbox for TLDs must be integratable into the registry and registrar systems. The requirements to achieve that are missing.