Mapping Top-Level Domains' Past, Present and Future DNSSEC Status

Shinkuro, Inc.
October 1, 2013

Version 6, October 1, 2013

Table of Contents

Introduction

We believe there is value in having a coherent, comprehensive picture of the overall state of DNSSEC adoption by the world's top-level domains (TLDs). Such a picture would be the largest-scale reflection of various parties' progress in adopting DNSSEC in the past, present and future.

We have been examining how to track DNSSEC deployment in TLDs, and collecting data for this purpose, since October 2009. To generate a continually updated current best picture of DNSSEC adoption, we have created a database to store available information about the past, present and future DNSSEC status of the world's TLDs, based on both reports from human sources and traces detectable via direct observation.

We've also created translation software to turn country-code TLD (ccTLD) database entries into maps so that readers can see when a TLD adopted DNSSEC or when we believe it will do so in the future, as well as how confident we are about this information.

The following paper describes how data collection and collation work throughout our process.

Characterizing TLDs' Deployment Processes

After several attempts at characterizing the steps most TLD operators go through in their deployment process, we have settled on listing five states that describe sets of actions. Some of these states are simple indications or declarations of intent, while others are technological steps which leave traces that are measurable via direct observation. Each step builds on those that came before.

The five states are:

- *Experimental*. A TLD operator announces or is reported to have begun internal experiments with DNSSEC. This involves a definite commitment of resources and energy, and the development of knowledge within the registry.
- *Announced*. The TLD operator has publicly committed to deploying DNSSEC in the future. This is a political and organizational step rather than a technical development.
- *Partial*. This is the first state involving DNSSEC deployment in the live zone: The TLD signs its zone, and one visible indicator of this is that its SOA record is signed. The signed SOA record is the first observable, technical evidence that the TLD is deploying DNSSEC in its operational zone, but at least in principle, the appearance of other DNSSEC-related records is also possible. We regularly check live zones for signed SOA records and cross-check that information against the reports in our database. We can reconcile our observations of this partial deployment with others' measurements.
- *DS in the Root*. The TLD has signed its zone *and* a corresponding DS record has been added to the root. At this point the zone offers a chain of trust validatable

all the way to the root, and as previously, we can correlate our observations with others' measurements to verify this state.

- *Full (or Operational)*. The zone is signed, has a DS record in the root, *and* announces or reports that it accepts delegations generally. Subzones can be signed and add their DS records to the parent TLD.

Our database is continually evolving. We do not have information regarding DNSSEC deployment for all TLDs, and in many cases our database contains no information about a TLD.

Only the Partial and DS in the Root states are externally observable, and only after they've happened, meaning these are highly reliable indications that a TLD has deployed DNSSEC to one or the other extent. While DNSSEC is not usable in the Partial state, we have created this separate category so as to highlight the beginning of actual DNSSEC deployment and the message this sends about the TLD's progress toward full deployment. We expect that Partial is a temporary and transitional state en route to the DS in the Root and Full states.

What We Record

Each time we receive information indicating a TLD's state, we record five things about that information in our database:

- The zone's (TLD's) name
- One of the five operational states noted above
- The date at which the state began or will begin in the future
- The date at which we are creating the record of this state
- A confidence score indicating how accurate we believe the information is

We also have a field in each record for notes that might provide additional clarity on the information gathered.

Each database entry is an addition of information regarding a specific TLD-state pair. If we receive multiple pieces of information at once—for example, that a TLD will reach Partial deployment on X date and expects to have a DS record in the root on Y date—we enter these as separate records.

We look at and preserve reports concerning the past, present and future. This lets us not only look at what a TLD's state was in the past and what we believe it will be in the future, but what we *thought* that TLD's state would be by some then-future point, and whether or not that report was borne out by events. (Other surveys of TLD DNSSEC deployment generally only take a snapshot of the present, with perhaps a nod toward historical data.)

What We Look For

We gather information from several sources in order to make a judgment about whether DNSSEC deployment has occurred and how far along this process is.

Colloquially, we note whether the events that form our current best picture are of the highest confidence, are consistent with reality and, for events that are listed as having already happened, are reported after the fact.

Highest confidence. Reports about DNSSEC status come to us in forms that range from very reliable (traceable DNS activity directly indicating that a TLD has signed its zone) to questionable (news reports of unknown reliability that refer to the future). We have devised a scale for indicating the confidence we have in incoming information, and use these confidence numbers to proactively and retroactively request updates from TLDs.

Consistent with reality. When a report indicates that a TLD has deployed DNSSEC by signing its zone, or by signing its zone and adding a DS record to the root, there should be observable traces of this activity. In these cases we go beyond the report, regardless of our previous level of confidence in it, and attempt to observe signs of this activity via software on the network.

Reported after the fact. Our ultimate goal is to have all reports of DNSSEC deployment be in the past, since this would indicate that TLDs have all signed their zones and that this can be verified with high confidence.

Highest Confidence

Since we have imperfect knowledge of TLDs' states and intentions, and since the inputs we get will include perceptions, best guesses and intentions as well as hard facts, we also assign a confidence score to each piece of news we receive on a scale of 1–4. Those numbers have the following meanings, in order of decreasing confidence:

- A 4 indicates that we have observed the state operationally by querying the TLD zone and/or querying the root. A 4 can only apply to the Partial or DS in the Root deployment states, which are directly observable.
- We use a 3 to indicate reports from principals within TLDs, such as an official within the TLD's operation.
- A 2 indicates reports from other respectable sources.
- All other reports receive a 1 confidence score.

Taking all the above factors into account, direct observations of past events (e.g. Partial, DS in the Root or Operational states reported with confidence scores of 4) are most certain while reports of predicted events that have a low confidence score (e.g. a report of a future Experimental or Full state that has a confidence score of 1) are least certain.

Ideally we will eventually have information that is accurate (confidence scores ≥3) and concerns events in the past (i.e., the date at which we create the record is later than the date at which the event took place). In the meantime, the database contains an ever-evolving mixture of backward-looking, present and forward-looking information, mainly with confidence scores below 3.

Consistent with Reality

We constantly seek to improve the quality of our information by requesting more current information from TLD operators directly, or by waiting for events to occur and

then checking on them observationally. We principally use reports from people at TLDs to populate this database, but also run software that checks these reports against data from the TLDs and the root to detect discrepancies.

Except for correcting our own data-entry errors, we don't remove information from the database; we continually add records, and previous records are available for future queries in case we for some reason need them. If an estimate of when some event is supposed to occur changes, we enter a new record.

Reported After the Fact

If a report specifies a state for the future and that future date arrives, we do two things: immediately reset the date at which the state is supposed to occur to 1–6 months in the future depending on how authoritative the original source of this date was, lower this information's confidence score to 1, and then check with the TLD or via other reports as to whether the state has occurred. Thus, the anticipated state remains in the future until we acquire evidence that it has occurred.

For reports of any given combination of TLD and state, we prefer the most recent record, even if it's considered of lower confidence.

Mapping

We code TLDs by category to more easily reconcile our data with surveys conducted by others. Each TLD is either a generic TLD (gTLD) such as .com or .net; a country-code TLD (ccTLD), representing a nation-state or territory; a regional TLD that doesn't correspond to a single nation-state or territory; or one of the 11 experimental TLDs currently in the root zone that do not yet correspond to an actual TLD operation.

The distinction between ccTLD and gTLD is deeply ingrained in ICANN's processes. gTLDs operate under contract from ICANN, while ccTLDs are not; they fall within the framework of RFC 1591 and ultimately are governed by a specific government.

The four groups mentioned above—gTLDs, ccTLDs, regional TLDs and experimental TLDs—are further divided into "classic" TLDs that use the Latin alphabet and internationalized domain names (IDNs) that use non-Latin alphabets, as with Arabic, Chinese, Russian and others.

At present, we produce maps showing the deployment status of ccTLDs. We track exactly one ccTLD as the principal one for a given nation-state or territory (e.g., we consider .uk to be England's principal ccTLD rather than .gb).

While regional TLDs are similar to ccTLDs, they do not correspond as neatly to a single geographic entity, and we have not yet worked out display mechanisms for them. Russia continues to operate the old Soviet Union TLD, .su, and the European Union is represented by .eu. Both of these operate under ccTLD rules, so we treat them as a separate category.

Generic TLDs, IDNs and experimental TLDs are not amenable to representation on a map, and we do not yet have display mechanisms to accommodate them.

However, we do produce a table showing the numbers of TLDs in each of the eight combinations of gTLD/ccTLD/regional/experimental vs. classic/IDN that are in the Partial state or higher, and a similar table for those in the DS in the Root state or higher.

Reports We Generate

We generate maps and reports weekly. This involves the following actions:

- Survey the Internet to confirm or detect current Partial and DS in the Root states, and compare incoming data against what is already present in our database. These steps involve a Shinkuro-written program called dbvsoperation.py.
- A second, slightly different scan surveys TLDs to find out when a change in DNSSEC status occurred, again comparing it against any prediction or other information in our database. This step involves a Shinkuro-written program called Monitor_tld_dnssec.
- Check whether predictions have come true. Observable predictions are checked using the dbvsoperation.py software mentioned above. Determination of an Operational state, however, can only be confirmed by querying the TLD in question via email, and this is done by a human operator.
- Manage discrepancies by moving predictions further out into the future and sending queries to TLD managers. This is done by a human operator.

Following the above actions, a program that we have created called emailmapsandreports.py generates a weekly report on TLDs' status. It contains the following items:

Maps that look at present-day, year-ago and year-from-now ccTLD DNSSEC status. These maps are created using software described more fully in the Database Configuration

Generic x86 system (currently 32-bit Intel, but could be anything supporting the software below (Intel or AMD, 32- or 64-bit). It is configured with the software listed below.

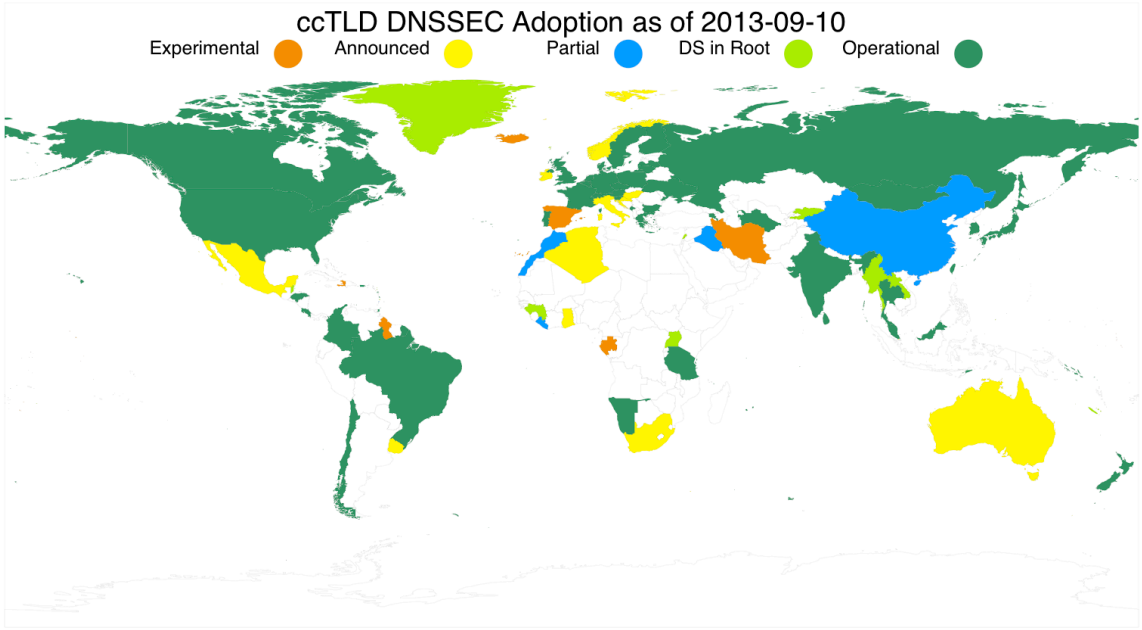
- Software Used section below, including Curl, Imagemagick, Natural Earth, Pyshp, Google Charts, dnssecmapcurlsbyregion.py, dnssecmapcurls.py and naturalearthmap.py.
- A key explaining the map legend
- CSV files that explain that day's DNSSEC status and properties for all TLDs
- A chart titled "TLDs--DS or Operational"
- Predictions of TLDs' DNSSEC status that have passed without authoritative speech or observation
- Predictions of future DNSSEC status based on information we have received

This report is emailed to a list of interested parties that is maintained by Shinkuro. On occasion, when enough time has passed and changes in the DNSSEC landscape are substantial enough to warrant it, we publish the maps as a blog entry on www.dnssec-deployment.org, sometimes generating an animated GIF showing DNSSEC progress over time. (At the time of this writing, one such blog post showed the sequence of ccTLD DNSSEC adoption starting at January 1, 2006 and running to July 1, 2015, based on information we have received.

We may also generate other reports as needed, such as ad-hoc reports based on SQL queries.

Example of Output

The following is an example of our regular weekly output: a map showing ccTLDs' DNSSEC status as of September 10, 2013, which is the culmination of the processes described up to this point:



Experimental -- Internal experimentaion announced or observed (9):	ES GA GY HT IR IS KI MS TO
Announced -- Public commitment to deploy (13):	AU DZ GH HK HR HU IE IT MX NO SG UY ZA
Partial -- Zone is signed but not in operation (no DS in root) (5):	CN IQ LR MA VC
DS in Root -- Zone is signed and its DS has been published (17):	AG BZ CC FO GI GL GN KG LA LB LC MM NC NU PW TV UG
Operational -- Accepting signed delegations and DS in root (61):	AC AM AT BE BG BR CA CH CL CO CR CX CZ DE DK FI FR GR GS HN IN IO JP KR LI LK LT LU LV ME MN MU MY NA NF NL NZ PL PM PR PT RE RU SB SC SE SH SI SX TF TH TL TM TT TW TZ UA UK US WF YT

Figure 1 – ccTLD DNSSEC Status at September 10, 2013

Technical Descriptions of the Database and Data Input

We frequently receive emailed reports from TLD operators concerning their zone's status. In addition, we actively query them when:

- We see visible changes based on whether our weekly scan of TLDs shows changes when compared with our database
- We see changes based on a separate, twice-daily query of the root zone that includes separate cryptographic analysis of TLD signatures
- Projected events are due to occur

If we have access to the TLD—as in the case of .com, for example—we can observe traces from it to tell whether it is at the DS in the Root phase or has passed into the Operational phase by accepting delegations from subdomains. If we do not have access to independent observations we must rely on the zone operator's accounts in making this determination.

Data is input to this system via a password-protected Web-based Xataface interface featuring pulldown menus, and data entered via this interface becomes entries in a MySQL database. The entire database, dumped as SQL text, is just over 400 KB in size.

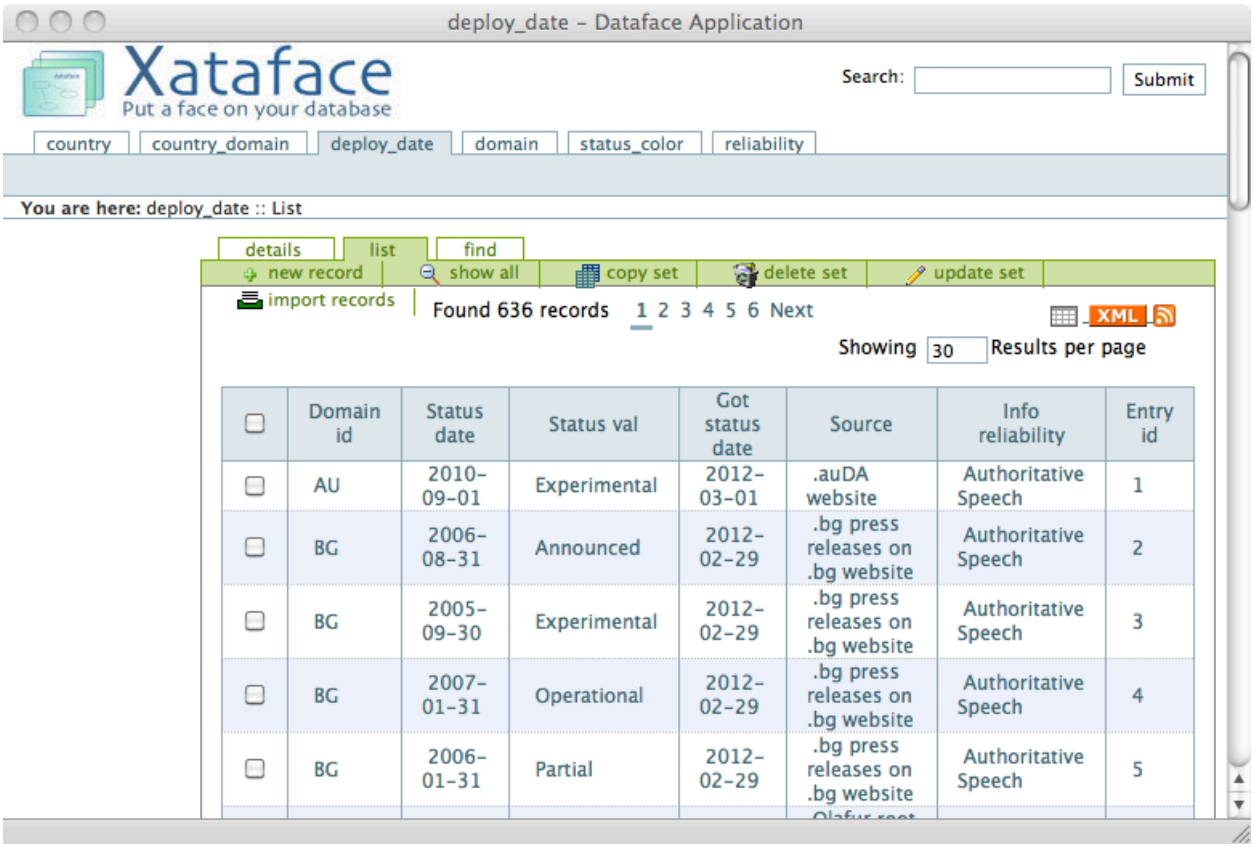


Figure 2 – Web interface for data entry

Database Architecture

The graphic below shows how input information flows into specific categories and can then be used to generate reports on all TLDs and to populate maps showing ccTLD DNSSEC status. New entries primarily deal with deployment dates since information in the database's other categories remains relatively static (a certain deployment category is always indicated by a specific color, the list of ccTLDs and regions we track remains relatively stable month to month, etc.).

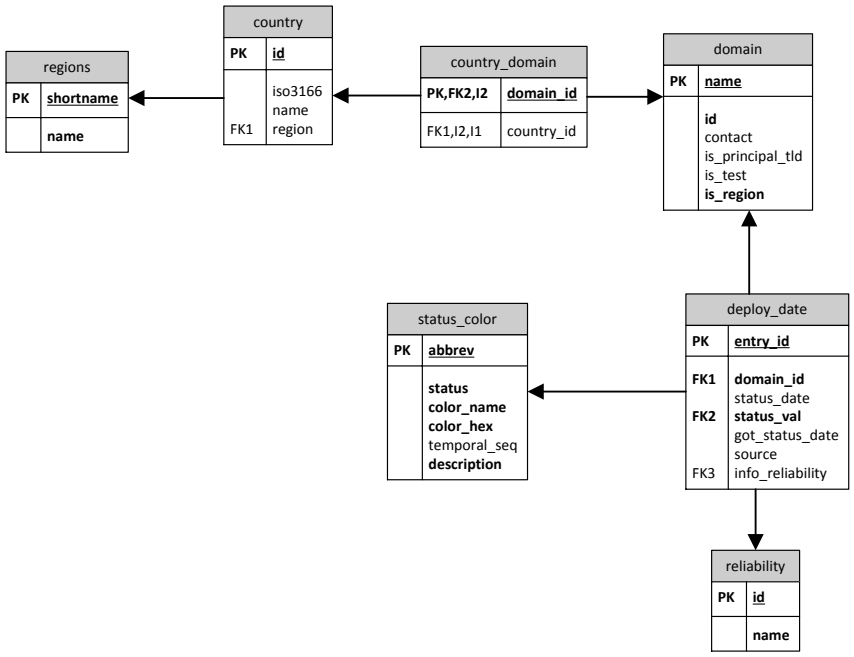


Figure 3 – Entity relationship name diagram for SQL

While we track the DNSSEC status of all TLDs, we only map ccTLDs since they are by far the most amenable to visual representation, i.e. on a map. We can quickly generate a complete text list of all other TLDs (generic, IDN and regional) and their DNSSEC status on demand, however.

New TLDs that come into existence from time to time are added based on information about them from IANA or ICANN. Since all new TLDs are now required to be DNSSEC-signed, the process of tracking these zones' DNSSEC status should become quite simple since they will have DS records in the root from their operational inception.

Database Configuration

Generic x86 system (currently 32-bit Intel, but could be anything supporting the software below (Intel or AMD, 32- or 64-bit). It is configured with the software listed below.

Software Used

Off-the-Shelf Software/Services

- LAMP stack (Linux/Apache/MySQL/PHP). Currently running on an Ubuntu 12.04 LTS system, but any modern Linux distribution (or something like FreeBSD) should work.
- Additional packages installed from the system package manager:
    - Subversion – Revision-control system to maintain software under development. In addition to the code, we use subversion to maintain a history of the database contents that have been dumped to SQL command streams via phpMyAdmin.
    - phpMyAdmin – Web GUI-based database administration.
    - Python 2.7.3 (often included in base system) – Programming language used for software described in Programs/Scripts Written by Shinkuro below.
    - Curl (often included in base system) – Command-line web tool for fetching images from the Google Charts API.
    - Imagemagick – Graphic manipulation tools used to string multiple maps together into an animated GIF.
    - Host (often included with base system or BIND tools installation) – Simple program for making DNS queries. Similar to, but less powerful than, dig.
- Software installed via external download:
    - Xataface – Web GUI for user data entry. Must be configured for the database and web server. See http://xataface.com.
    - Natural Earth – Data for plotting PDF maps. Shinkuro is using the 1:50,000,000 data set. See www.naturalearthdata.com.
    - ReportLab – Python software for generating PDF. See www.reportlab.com.
    - Pyshp – Library for reading Natural Earth shape files. See http://code.google.com/p/pyshp/.
- External services used:
    - Google Charts map – Web service for taking data and generating low-resolution PNG maps. Documented at https://developers.google.com/chart/. The maps API has been deprecated, but Google has not removed it.

Programs/Scripts Written by Shinkuro

- ccTLDmailmerge.py – Used once to generate custom email to TLD contacts asking for updated information on DNSSEC adoption.
- cctldstatusovertime.py – Prints status of all ccTLDs for which we have data for today and the same day in each of the past four years.

- currentcctldstatus.py – Prints current status of all primary (non-IDN) ccTLDs for which we have data.
- currenttldstatus.py – Prints current status of all TLDs for which we have data.
- dbvsobservation.py – Compares current status in database with what's visible in the root zone and from TLD nameservers.
- dnssecmapcurlsbyregion.py – Generates curl commands to obtain world or regional maps over time via the Google Charts API.
- dnssecmapcurls.py – Generates curl commands to obtain world maps over time via the Google Charts API.
- emailmapsandreports.py – Generates the weekly email that includes PNG maps and spreadsheets. Can also be run manually.
- Monitor_tld_dnssec – Scans TLDs to determine time at which DNSSEC status changed. Compares this information against our database.
- naturalearthmap.py – Generates a detailed PDF adoption map based on current data.

Appendix: MySQL Database Schema

```
-- Database: `dnssecmap`

-- Table structure for table `country`

CREATE TABLE IF NOT EXISTS `country` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `iso3166` char(2) DEFAULT NULL,
  `name` text,
  `region` varchar(10) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
);

-- Table structure for table `country_domain`

CREATE TABLE IF NOT EXISTS `country_domain` (
  `domain_id` bigint(20) NOT NULL DEFAULT '0',
  `country_id` bigint(20) DEFAULT NULL,
  PRIMARY KEY (`domain_id`),
  UNIQUE KEY `c_d_domainonly_uidx` (`domain_id`),
  UNIQUE KEY `c_d_country_domain_uidx` (`country_id`,`domain_id`)
);

-- Table structure for table `deploy_date`

CREATE TABLE IF NOT EXISTS `deploy_date` (
  `domain_id` bigint(20) NOT NULL DEFAULT '0',
  `status_date` date DEFAULT NULL,
  `status_val` char(3) NOT NULL DEFAULT '',
  `got_status_date` date DEFAULT NULL,
  `source` text,
  `info_reliability` int(11) DEFAULT NULL,
  `entry_id` bigint(10) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`entry_id`)
);

-- Table structure for table `domain`

CREATE TABLE IF NOT EXISTS `domain` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(30) CHARACTER SET ascii NOT NULL DEFAULT '',
  `contact` text,
  `is_principal_tld` tinyint(1) DEFAULT '0',
  `is_test` tinyint(1) DEFAULT '0',
  `is_region` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`name`),
```

```
  UNIQUE KEY `id` (`id`)
);

-- Table structure for table `regions`

CREATE TABLE IF NOT EXISTS `regions` (
  `name` varchar(40) NOT NULL,
  `shortname` varchar(10) NOT NULL,
  PRIMARY KEY (`shortname`)
);

-- Table structure for table `reliability`

CREATE TABLE IF NOT EXISTS `reliability` (
  `id` int(8) unsigned NOT NULL,
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`id`)
);

-- Table structure for table `status_color`

CREATE TABLE IF NOT EXISTS `status_color` (
  `abbrev` char(3) NOT NULL,
  `status` text,
  `color_name` text,
  `color_hex` text,
  `temporal_seq` int(11) DEFAULT NULL,
  `description` text NOT NULL,
  PRIMARY KEY (`abbrev`),
  UNIQUE KEY `abbrev` (`abbrev`)
);
```