Portia Documentation

Release 1.0.0

Praekelt Foundation

July 27, 2017

Contents

1 Installation **3**

2 Importing database files **5**

3 Running the web server **7**
3.1 Resolving 7
3.2 Querying 8
3.3 Annotating 8

4 Running the TCP socket server **9**
4.1 Get 9
4.2 Annotate 9
4.3 Resolve 10

A small service that allows one to:

- lookup phone numbers in number porting databases.
- annotate records with extra information when needed.

CHAPTER 1

Installation

First make sure you have Redis installed and running:

```
$ apt-get install redis-server
```

```
$ virtualenv ve
$ source ve/bin/activate
(ve)$ pip install portia
(ve)$ portia --help
```

CHAPTER 2

Importing database files

```
(ve)$ portia import porting-db path/to/file.csv
```

CHAPTER 3

Running the web server

```
(ve)$ portia run
2015-10-16 18:22:35+0200 [-] Log opened.
2015-10-16 18:22:35+0200 [-] Starting factory <txredisapi.RedisFactory instance at 0x105b3bea8>
2015-10-16 18:22:35+0200 [HiredisProtocol,client] Site starting on 8000
2015-10-16 18:22:35+0200 [-] Starting factory <twisted.web.server.Site instance at 0x105b6ef80>
```

By default this will listen on `localhost:8000`.

Resolving

Resolving an MNO can be done via the `/resolve` endpoint:

```
$ curl localhost:8000/resolve/27761234567
{
   "entry": {
     "ported-to-timestamp": "2015-10-16T19:26:41.943293",
     "observed-network-timestamp": "2015-10-16T19:49:21.130930",
     "ported-to": "CELLC",
     "observed-network": "MTN"
   },
   "network": "MTN",
   "strategy": "observed-network"
 }
```

The `network` key has the most likely network the MSISDN is homed on. The `strategy` is the strategy used to make that decision. The `strategy` is currently very naïve, it gets the most recent `observed-network` or `ported-to` timestamp and returns that.

If all else fails it falls back to guessing based on the prefix:

```
$ curl localhost:8000/resolve/27760000000
{
  "entry": {},
  "network": "VODACOM",
  "strategy": "prefix-guess"
}
```

Querying

Looking up everything for a known phone number:

```
$ curl http://localhost:8000/entry/27123456780
{
  "ported-to-timestamp": "2015-10-11T00:00:00",
  "ported-from": "MNO1",
  "ported-to": "MNO2",
  "ported-from-timestamp": "2015-10-11T00:00:00"
}
```

Looking up a single key for a phone number:

```
$ curl http://localhost:8000/entry/27123456780/ported-to
{
  "ported-to": "MNO2",
  "ported-to-timestamp": "2015-10-11T00:00:00"
}
```

Annotating

Portia has a number of defined annotations that it supports, these are:

- observed-network
- ported-from
- ported-to
- do-not-call

Adding an observed network annotation for a phone number:

```
$ curl -XPUT -d MNO3 http://localhost:8000/entry/27123456780/observed-network
"MNO3"

$ curl http://localhost:8000/entry/27123456780/observed-network
{
  "ported-to": "MNO2",
  "ported-to-timestamp": "2015-10-11T00:00:00",
  "ported-from": "MNO1",
  "ported-from-timestamp": "2015-10-11T00:00:00",
  "observed-network": "MNO3",
  "observed-network-timestamp": "2015-10-13T06:54:18.797250"
}
```

Custom annotations are allowed if the key is prefixed with `X-`:

```
$ curl -XPUT -d bar http://localhost:8000/entry/27123456780/X-foo
"bar"
```

CHAPTER 4

Running the TCP socket server

```
(ve)$ portia run --tcp
2015-10-16 18:22:51+0200 [-] Log opened.
2015-10-16 18:22:51+0200 [-] Starting factory <txredisapi.RedisFactory instance at 0x10a969ea8>
2015-10-16 18:22:51+0200 [HiredisProtocol,client] JsonProtocolFactory starting on 8001
2015-10-16 18:22:51+0200 [-] Starting factory <portia.protocol.JsonProtocolFactory instance at 0x10a9
2015-10-16 18:22:51+0200 [HiredisProtocol,client] Site starting on 8000
2015-10-16 18:22:51+0200 [-] Starting factory <twisted.web.server.Site instance at 0x10a9a03b0>
```

By default this will listen on `localhost:8001`. You can specify a different endpoint with `--tcp-endpoint=tcp:8080:interface=127.0.0.1` as an example

JSON is used for the socket protocol. It uses `\r\n` as a delimiter

Note: The timestamp values are all in ISO 8601 format. Timezone naive timestamps are assumed to be in UTC and will be stored internally as such.

Get

```
$ telnet localhost 8001
> {"cmd": "get", "id": 1, "version": "0.1.0", "request": {"msisdn": "27761234567"}}
< {"status": "ok", "cmd": "reply", "version": "0.1.0", "reference_id": 1, "response": {"ported-to-tir
```

Annotate

```
$ telnet localhost 8001
> {"cmd": "annotate", "id": 2, "version": "0.1.0", "request": {"msisdn": "27761234567", "key": "X-Foo
< {"status": "ok", "cmd": "reply", "version": "0.1.0", "reference_id": 2, "response": "OK", "referenc

$ telnet localhost 8001
> {"cmd": "get", "id": 3, "version": "0.1.0", "request": {"msisdn": "27761234567"}}
< {"status": "ok", "cmd": "reply", "version": "0.1.0", "reference_id": 3, "response": {"ported-to-tir
```

Resolve

```
$ telnet localhost 8001
> {"cmd": "resolve", "id": 4, "version": "0.1.0", "request": {"msisdn": "27761234567"}}
< {"status": "ok", "cmd": "reply", "version": "0.1.0", "reference_id": 4, "response": {"entry": {"por
```