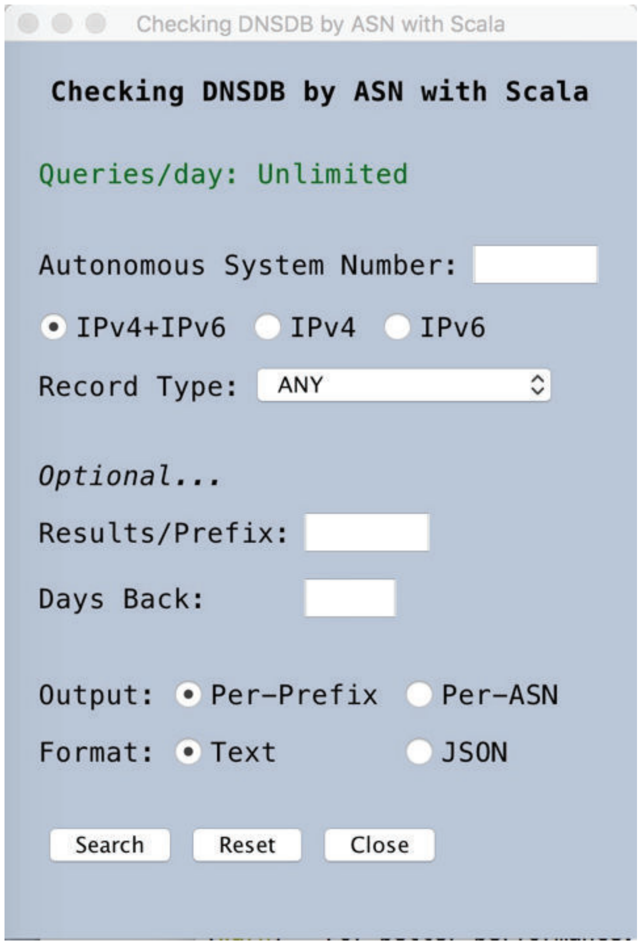
FARSIGHT SECURITY

White Paper

Checking DNSDB By ASN (ASN --> Prefixes --> Domain Names) Another Demonstration Scala Project

v1.0, May 24, 2017
Joe St Sauver, Ph.D. (stsauver@fsi.io)


Checking DNSDB by ASN with Scala

Checking DNSDB by ASN with Scala

Queries/day: Unlimited

Autonomous System Number:

(● IPv4+IPv6) ○ IPv4 ○ IPv6

Record Type: ANY

Optional...

Results/Prefix:

Days Back:

Output: (● Per-Prefix) ○ Per-ASN

Format: (● Text) ○ JSON

Table of Contents

1. Introduction

Farsight Security customers often lookup IP addresses (such as 128.223.32.35) or CIDR prefixes (such as 128.223.0.0/16) in DNSDB. When users do that, DNSDB returns the RRnames that have been seen associated with those addresses.

At times, a user might be interested not just in a handful of addresses or CIDR prefixes, but ALL the CIDR prefixes originated by a particular autonomous system number (or "ASN").

Backfill: What's An ASN?

Just to ensure that everyone's up to speed on the jargon in this white paper, an "autonomous system number" or "ASN" is usually technically defined as a number assigned to a group of network addresses, managed by a particular network operator, sharing a common routing policy. Most ISPs, large corporations, and university networks have an ASN. For example, Google uses AS15169, Sprint uses AS1239, Intel uses AS4983, the University of California at Berkeley uses AS25 and so on.

Some large networks with particularly complex routing policies may have more than one ASN; others, with simple routing policies and only a single upstream network provider, may have none (their network blocks are announced to the Internet using their upstream provider's ASN).

Bottom line, in general, think of an autonomous system number as a number that "maps to" or represents a particular provider or network. As such, it is a useful way to aggregate and sort IP addresses into useful chunks, even though its initial purpose (and continued most important usage) is in conjunction with BGP for inter-AS routing of network traffic.

Today we will create an application that will:

1) Get an ASN from the user

2) Map that ASN to a set of IPv4 and/or IPv6 prefixes (based on BGP routing data from BGPview.io)

3) Do a DNSDB Rdata query for each IPv4 or IPv6 prefix found in step 2

But how many prefixes might an ASN originate?

Some ASNs might have only one prefixes; others might have thousands.

An ASN that isn't being used, or isn't being used on the public Internet, might not have any publicly-known prefixes at all.

2. An Example ASN: AS3582 (University of Oregon)

Consider AS3582, the autonomous system number used by the University of Oregon.

If we check http://bgp.he.net, we can see that AS3582 announces 4 IPv4 prefixes and 2 IPv6 prefixes, see:

```
http://bgp.he.net/AS3582#_prefixes

     128.223.0.0/16
     163.41.128.0/17
     184.171.0.0/17
     207.98.72.0/21

http://bgp.he.net/AS3582#_prefixes6

     2001:468:d01::/48
     2607:8400::/32
```

If we wanted to check DNSDB for the rrnames associated with each of those prefixes, we could do six DNSDB rdata queries, one for each of the address prefixes. Because there are only six prefixes in this case, that's not a very onerous chore.

However, when we have hundreds or even thousands of prefixes, as some large networks do, we'd really prefer to maintain a higher level of abstraction and just refer to that set of prefixes by the autonomous system number that's originating them. Mentioning one AS number is a LOT more convenient than enumerating a large number of specific prefixes.

3. Mapping An ASNs to A Prefix List Using the BGPView API, curl, jq and sed

While consulting the http://bgp.he.net web site works great for interactive use, when writing an application, we need an API that can give us the relevant list of prefixes, instead.

BGPView[1] is one service that lets folks map an ASN to its associated list of prefixes programmatially, returning the routing information for an ASN as a readily-parseable JSON object. For example, to retrieve the prefixes for AS3582, we could use **curl:**[2]

```
$ curl -s "https://api.bgpview.io/asn/3582/prefixes"
```

That command would generate output that looks like (wrapped):

```
{"status":"ok","status_message":"Query was successful","data":{"ipv4_prefixes":
[{"prefix":"128.223.0.0\/16","ip":"128.223.0.0","cidr":16,"roa_status":"None","name":
"UONET","description":"University of Oregon","country_code":"US","parent":{"prefix":
"128.223.0.0\/16","ip":"128.223.0.0","cidr":16,"rir_name":"ARIN","allocation_status":
"unknown"}},{"prefix":"163.41.128.0\/17","ip":"163.41.128.0","cidr":17,"roa_status":
"None","name":"UOFORESEARPK","description":"University of Oregon","country_code":
"US","parent":{"prefix":"163.41.0.0\/16","ip":"163.41.0.0","cidr":16,"rir_name":
"ARIN","allocation_status":"unknown"}},{"prefix":"184.171.0.0\/17","ip":"184.171.0.0",
"cidr":17,"roa_status":"None","name":"UONET","description":"University of Oregon",
"country_code":"US","parent":{"prefix":"184.171.0.0\/17","ip":"184.171.0.0","cidr":
17,"rir_name":"ARIN","allocation_status":"unknown"}},{"prefix":"207.98.72.0\/21","ip":
```

[1] https://bgpview.io/

[2] https://curl.haxx.se/

```
"207.98.72.0","cidr":21,"roa_status":"None","name":"OREGON-EXCH","description":
"Oregon Exchange","country_code":"US","parent":{"prefix":"207.98.0.0\/17","ip":
"207.98.0.0","cidr":17,"rir_name":"ARIN","allocation_status":"unknown"}}],
"ipv6_prefixes":[{"prefix":"2001:468:d01::\/48","ip":"2001:468:d01::","cidr":48,
"roa_status":"None","name":"INTERNET2-OREGON-V6","description":"Oregon GigaPOP",
"country_code":"US","parent":{"prefix":"2001:468::\/32","ip":"2001:468::","cidr":
32,"rir_name":"ARIN","allocation_status":"unknown"}},{"prefix":"2607:8400::\/32",
"ip":"2607:8400::","cidr":32,"roa_status":"None","name":"UONET","description":
"University of Oregon","country_code":"US","parent":{"prefix":"2607:8400::\/32","ip":
"2607:8400::","cidr":32,"rir_name":"ARIN","allocation_status":"unknown"}}]},
"@meta":{"time_zone":"UTC","api_version":1,"execution_time":"5.24 ms"}}
```

That's just a ***little*** more information than we need for our particular purposes!

Fortunately, at the Unix command prompt, we can use **jq**[3] and **sed**[4] to filter out all the bits we don't care about:

```
$ curl -s "https://api.bgpview.io/asn/3582/prefixes" | jq '.data.ipv4_prefixes[].prefix'
| sed 's/"//g'

     128.223.0.0/16
     163.41.128.0/17
     184.171.0.0/17
     207.98.72.0/21

$ curl -s "https://api.bgpview.io/asn/3582/prefixes" | jq '.data.ipv6_prefixes[].prefix'
| sed 's/"//g'

     2001:468:d01::/48
     2607:8400::/32
```

That approach works fine if users are comfortable scripting, but let's create an actual program that doesn't require any level of Unix command line comfort.

4. Selecting A JSON Parser/Deserializer Library for Scala

Since we've previously shown how Scala could be used to create a demo graphical user interface to DNSDB API, let's extend that Scala code for to create a Scala tool that will let us search DNSDB API by ASN.

While much of what we did in the earlier article is directly applicable, we still need to pick a library to help us parse (or "deserialize") the JSON format data that the BGPview API returns.

[3] https://stedolan.github.io/jq/
[4] https://www.gnu.org/software/sed/manual/sed.html

Scala has a plethora of JSON parsing and formating options available, including (in alphabetical order):

- akka-http-spray-json[5]
- argonaut[6]
- circe[7]
- jackson-module-scala[8]
- jawn[9]
- json4s[10]
- lift-json[11]
- Play Framework's ScalaJson[12]
- Rapture JSON[13]
- Scala's own built-in JSON parsing utility library[14]
- sjson[15]
- sphere-json[16]

Because of the quality of its documentation and its broad adoption, we'll arbitrarily pick the Play Framework's ScalaJSON routines to use to parse the api.bgpviewe.io data.

Before adapting our previous code to the ASN-based case, let's work on a small example that simply parses a small saved sample of api.bgpview.io data. Sample Scala code to do this is included as Appendix 1.

To compile and run that example code, assuming you've already installed Scala and sbt as previously described in our earlier whitepaper, you'd do:

```
$ mkdir JsonTest
$ cd JsonTest

$ mkdir -p src/{main,test}/{java,resources,scala}
$ mkdir lib project target

$ nano src/main/scala/JsonTest.scala
copy and paste Appendix 1 here
<ctrl-x><ctrl-o>

$ nano build.sbt
copy and paste Appendix 2 here
<ctrl-x><ctrl-o>

$ sbt run
```

[5] http://doc.akka.io/docs/akka-http/current/scala/http/common/json-support.html); spray-json itself is no longer maintained (see the note on http://spray.io/)
[6] http://argonaut.io/
[7] https://circe.github.io/circe/
[8] https://github.com/FasterXML/jackson-module-scala
[9] https://github.com/non/jawn
[10] http://json4s.org/
[11] https://github.com/lift/framework/tree/master/core/json
[12] https://www.playframework.com/documentation/2.4.x/ScalaJson
[13] https://github.com/propensive/rapture/blob/dev/doc/json.md
[14] http://www.scala-lang.org/api/rc2/scala/util/parsing/json/package.html
[15] https://github.com/debasishg/sjson
[16] https://github.com/sphereio/sphere-scala-libs/tree/master/json

```
[...]
[info] Running JsonTest
128.223.0.0/16
163.41.128.0/17
184.171.0.0/17
207.98.72.0/21
2001:468:d01::/48
2607:8400::/32
```

5. Building An ASN-ified Version of Our Earlier GUI application

Now that we have working Scala code to parse the list of prefixes associated with an ASN, we're finally ready to modify our earlier GUI Scala application to look up prefixes-by-ASN using the DNSDB API.

The program we've written offer users:

1) The option of investigating IPv4 *and* IPv6 prefixes under an ASN, or just IPv4 or just IPv6 prefixes
2) The ability to just select specific record types (note: some record types may not make sense for rdata IP queries, e.g., you shpild never see CNAME records because CNAME records don't have IP addresses)
3) A chance to limit the maximum number of results returned per prefix
4) Time fencing, to results to just the most recent *N* days
5) The ability to save output in either of two formats:
    a) **Per-Prefix,** with results for each prefix in an individual file under a per-ASN directory, or
    b) **Per-ASN,** with all results aggregated into a single file named for the ASN
6) DNSDB Text or JSON output

Because this application may potentially generate hundreds or even *thousands* of output files, we won't pop up a seperate alert notice for each file we create, we'll just log the prefixes we look up and the location of the output to the terminal window from which we launch our application.

As noted in point 5) above, output files may be in either of two formats. To explain how output is set up, it may help to look at an example of per-prefix output, and an example of per-ASN output:

```
dnsdb-output
└── 2017-04-23
    ├── 200959-AS87         <-- Per-Prefix Output under an ASN subdirectory
    │   ├── 200959-129.79.0.0,16-ANY.text
    │   ├── 201000-134.68.0.0,16-ANY.text
    │   ├── 201003-140.182.0.0,16-ANY.text
    │   ├── 201006-140.182.0.0,19-ANY.text
    │   ├── 201007-149.159.0.0,16-ANY.text
    │   ├── 201009-149.160.0.0,14-ANY.text
    │   ├── 201020-149.160.128.0,17-ANY.text
   [etc...]
```

versus:

```
dnsdb-output
└── 2017-04-23
    └── 201043-AS101-ANY.text  <-- A single aggregated Per-ASN file (all results)
```

FAQ: "What are the numbers in the above output, like "2017-04-23" and 201043?"

Answer: That's a date (in YYYY-MM-DD format) and a timestamp (in HHMMSS format), representing the time the query was run.

6. Building the Per-ASN Application

In order to build and run our per-ASN application, we'll need to have Scala, SBT and Java installed as previously described. You'll also need to have a Farsight DNSDB API key installed in $HOME/.dnsdb-apikey.txt (if you don't have a DNSDB API key, please see https://www.farsightsecurity.com/order-services/ for assistance.)

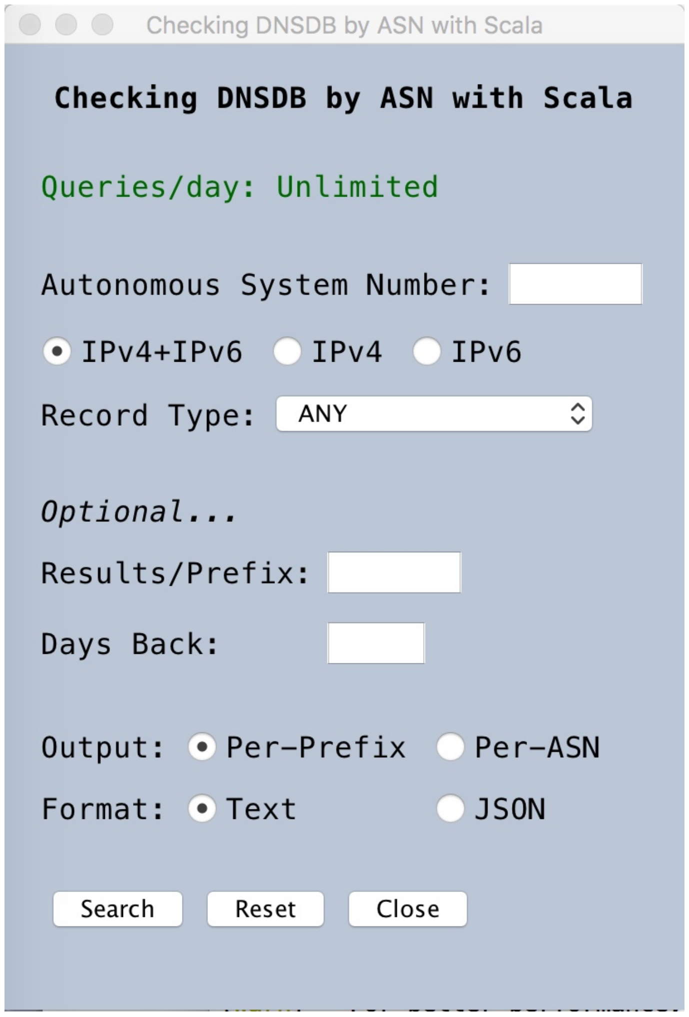
Now create the following directory structure for the project:

```
$ mkdir CheckASN
$ cd CheckASN

$ mkdir -p src/{main,test}/{java,resources,scala}
$ mkdir lib project target

$ nano src/main/scala/CheckASN.scala
copy and paste Appendix 3 here
<ctrl-x><ctrl-o>

[now use nano to create the other files shown in Appendix 4-8]
```

Now, from the top level CheckASN project home directory, compile and run the application by saying:

```
$ sbt run
```

You'll then see a window that looks like:


Checking DNSDB by ASN with Scala

Checking DNSDB by ASN with Scala

Queries/day: Unlimited

Autonomous System Number:

[✓ IPv4+IPv6] IPv4 IPv6

Record Type: ANY

Optional...

Results/Prefix:

Days Back:

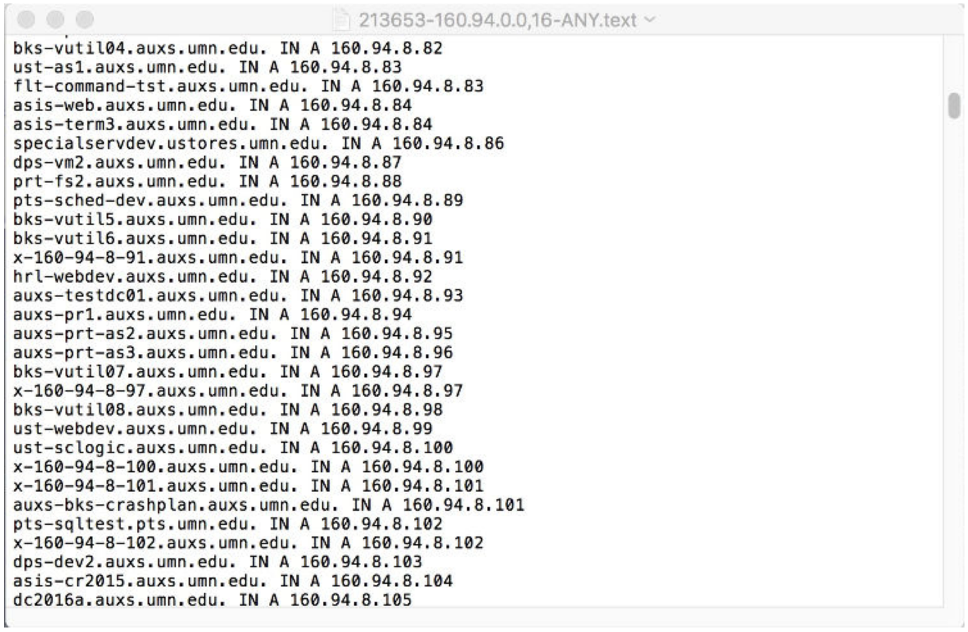
Output: [✓ Per-Prefix] Per-ASN

Format: [✓ Text] JSON

In the simplest of cases, you can then plugin an ASN and click the Search button. For example, the University of Minnesota uses AS217. Plugging in 217 and hitting Search, we see:

```
[info] Prefix is = 128.101.0.0/16
[info] Making subdiredctory/Users/joe/dnsdb-output/2017-04-23/213648-AS217
[info] Output is going to = /Users/joe/dnsdb-output/2017-04-23/213648-AS217/213648-
128.101.0.0,16-ANY.text
[info]
[info] Prefix is = 131.212.0.0/16
[info] Output is going to = /Users/joe/dnsdb-output/2017-04-23/213648-AS217/213650-
131.212.0.0,16-ANY.text
[info]
[info] Prefix is = 134.84.0.0/16
[info] Output is going to = /Users/joe/dnsdb-output/2017-04-23/213648-AS217/213651-
134.84.0.0,16-ANY.text
[info]
[info] Prefix is = 146.57.0.0/16
[info] Output is going to = /Users/joe/dnsdb-output/2017-04-23/213648-AS217/213652-
146.57.0.0,16-ANY.text
[info]
[info] Prefix is = 160.94.0.0/16
[info] Output is going to = /Users/joe/dnsdb-output/2017-04-23/213648-AS217/213653-
160.94.0.0,16-ANY.text
[info]
[info] Prefix is = 2001:470:22::/48
[info] Output is going to = /Users/joe/dnsdb-output/2017-04-23/213648-AS217/213654-
2001#470#22##,48-ANY.text
```

Looking at one of those output files by clicking it in the Mac Finder, we see output that looks like:



```
bks-vutil04.auxs.umn.edu. IN A 160.94.8.82
ust-as1.auxs.umn.edu. IN A 160.94.8.83
flt-command-tst.auxs.umn.edu. IN A 160.94.8.83
asis-web.auxs.umn.edu. IN A 160.94.8.84
asis-term3.auxs.umn.edu. IN A 160.94.8.84
specialservdev.ustores.umn.edu. IN A 160.94.8.86
dps-vm2.auxs.umn.edu. IN A 160.94.8.87
prt-fs2.auxs.umn.edu. IN A 160.94.8.88
pts-sched-dev.auxs.umn.edu. IN A 160.94.8.89
bks-vutil5.auxs.umn.edu. IN A 160.94.8.90
bks-vutil6.auxs.umn.edu. IN A 160.94.8.91
x-160-94-8-91.auxs.umn.edu. IN A 160.94.8.91
hrl-webdev.auxs.umn.edu. IN A 160.94.8.92
auxs-testdc01.auxs.umn.edu. IN A 160.94.8.93
auxs-pr1.auxs.umn.edu. IN A 160.94.8.94
auxs-prt-as2.auxs.umn.edu. IN A 160.94.8.95
auxs-prt-as3.auxs.umn.edu. IN A 160.94.8.96
bks-vutil07.auxs.umn.edu. IN A 160.94.8.97
x-160-94-8-97.auxs.umn.edu. IN A 160.94.8.97
bks-vutil08.auxs.umn.edu. IN A 160.94.8.98
ust-webdev.auxs.umn.edu. IN A 160.94.8.99
ust-sclogic.auxs.umn.edu. IN A 160.94.8.100
x-160-94-8-100.auxs.umn.edu. IN A 160.94.8.100
x-160-94-8-101.auxs.umn.edu. IN A 160.94.8.101
auxs-bks-crashplan.auxs.umn.edu. IN A 160.94.8.101
pts-sqltest.pts.umn.edu. IN A 160.94.8.102
x-160-94-8-102.auxs.umn.edu. IN A 160.94.8.102
dps-dev2.auxs.umn.edu. IN A 160.94.8.103
asis-cr2015.auxs.umn.edu. IN A 160.94.8.104
dc2016a.auxs.umn.edu. IN A 160.94.8.105
```

7. Packaging

If desired, you can package this application as demonstrated in the previous white paper, thereby eliminating the need to use sbt to run the application. Packaging the application also facilitates distributing the application to 3rd-party users. See the first white paper in this series for details.

8. Considerations

a) This is just a demonstration/trial application meant to illustrate how you can develop your own interface to DNSDB API. This demo application is NOT a new production/company-supported DNSDB query interface!

b) The demo application is intentionally **NOT configured to run parallel DNSDB queries.**

c) While the application uses SSL/TLS, unlike our first Scala example, **we are NOT forcing the application to use particularly strong cryptography.** It should run fine with either the default crypto policy or with the unlimited strength crypto policy files.

d) If you do decide to try building and running this application, please note that **you can quickly burn through a lot of queries by investigating ASNs that announce a large number of prefixes.** Please ensure you don't accidentally exhaust your entire daily DNSDB query allocation by just making a couple of large test "by-ASN" queries!

If you need to get a sense of how many prefixes a given query might "eat," try searching http://bgp.he.net/ for an organization of interest, such as "Florida State University". You'll see that Florida State actually uses multiple ASNs, and each of those ASNs has multiple prefixes:

```
ASN          Prefixes Originated
-------------------------------
AS2553           6
AS36843          10
```

Other ASNs have many more prefixes. Examples of large(r) ASNs can be found at http://www.cidr-report.org/as2.0/ or by browsing around http://bgp.he.net. For example:

ASN	Organization	Prefixes Originated
AS4538	China Education and Research Network Center	5,577
AS3408	Akamai International B.V.	3,408
AS3356	Level 3 Communications, Inc.	1,611
AS701	Verizon Business/UUNet	1,588
AS1702	AT&T Services, Inc.	1,502

Most ASNs will tend to be far smaller.

9. Acknowledgement

We'd like to thank BGPview for making their BGPview API available to the community.

10. Conclusion

You've now seen another simple example of how you can use Scala and supporting libraries to query DNSDB.

Maybe it's time for you to try your hand at coding an application that queries DNSDB?

Appendix 1. Sample Scala Code Demonstrating Extraction of Prefixes For An ASN From Saved BGPview Results (JsonTest/src/main/scala/JsonTest.scala)

```
import scala._
import play.api.libs.json._

object JsonTest extends App {
  // sample JSON for testing
  // created with: $ curl "https://api.bgpview.io/asn/3582/prefixes" > foo.txt
  // then cut and pasted in here (note that this data is wrapped for display here!)

  var rawJsonLines = """
{"status":"ok","status_message":"Query was successful","data":{"ipv4_prefixes":
[{"prefix":"128.223.0.0\/16","ip":"128.223.0.0","cidr":16,"roa_status":"None",
"name":"UONET","description":"University of Oregon", "country_code":"US","parent":
{"prefix":"128.223.0.0\/16","ip":"128.223.0.0","cidr":16,"rir_name":"ARIN",
"allocation_status":"unknown"}},{"prefix":"163.41.128.0\/17","ip":"163.41.128.0",
"cidr":17,"roa_status":"None","name":"UOFORESEARPK","description":
"University of Oregon","country_code":"US","parent":{"prefix":"163.41.0.0\/16",
"ip":"163.41.0.0","cidr":16,"rir_name":"ARIN","allocation_status":"unknown"}},
{"prefix":"184.171.0.0\/17","ip":"184.171.0.0","cidr":17,"roa_status":"None",
"name":"UONET","description":"University of Oregon","country_code":"US","parent":
{"prefix":"184.171.0.0\/17","ip":"184.171.0.0","cidr":17,"rir_name":"ARIN",
"allocation_status":"unknown"}},{"prefix":"207.98.72.0\/21","ip":"207.98.72.0","cidr":
21,"roa_status":"None","name":"OREGON-EXCH","description":"Oregon Exchange",
"country_code":"US","parent":{"prefix":"207.98.0.0\/17","ip":"207.98.0.0","cidr":
17,"rir_name":"ARIN","allocation_status":"unknown"}}],"ipv6_prefixes":[{"prefix":
"2001:468:d01::\/48","ip":"2001:468:d01::","cidr":48,"roa_status":"None","name":
"INTERNET2-OREGON-V6","description":"Oregon GigaPOP","country_code":"US","parent":
{"prefix":"2001:468::\/32","ip":"2001:468::","cidr":32,"rir_name":"ARIN",
"allocation_status":"unknown"}},{"prefix":"2607:8400::\/32","ip":"2607:8400::","cidr":
32,"roa_status":"None","name":"UONET","description":"University of Oregon",
"country_code":"US","parent":{"prefix":"2607:8400::\/32","ip":"2607:8400::",
"cidr":32,"rir_name":"ARIN","allocation_status":"unknown"}}]},"@meta":{"time_zone":
"UTC","api_version":1,"execution_time":"4.57 ms"}}
"""

// The JSON format unfortunately uses "prefix" as a label for the parent
// prefixes AS WELL AS the raw prefixes we're interested in; let's rename
// the parent prefixes to simplify later processsing (kludge: true)

rawJsonLines = rawJsonLines.replace("\"parent\":{\"prefix\"",
                                    "\"parent\":{\"parent_prefix\"")

// parse and print the JSON
  val elements = Json.parse(rawJsonLines)
  var my_ipv4_prefixes = (elements \ "data" \ "ipv4_prefixes" \\ "prefix")
  var my_ipv6_prefixes = (elements \ "data" \ "ipv6_prefixes" \\ "prefix")

  for (e <- my_ipv4_prefixes) println(e.toString.replace("\"",""))
  for (e <- my_ipv6_prefixes) println(e.toString.replace("\"",""))
}
```

Appendix 2. JsonTest/build.sbt for JsonTest

```
name := "JsonTest"

version := "1.0"

scalaVersion := "2.12.1"

// https://github.com/playframework/play-json
libraryDependencies += "com.typesafe.play" %% "play-json" % "2.6.0-M1"
```

Appendix 3. CheckASN/src/main/scala/CheckASN.scala

```
package checkasn

import java.awt._
import java.awt.Component._
import java.io._
import java.lang._
import java.net._
import java.text._
import java.util._
import javax.swing.JFrame._
import play.api.libs.json._
import scala._
import scala.io._
import scala.swing._
import swing._
import Swing._

class UI extends MainFrame {
  // Our main window. Closing it will kill the application

  // ================================================================
  // Definitions

  title = "Checking DNSDB by ASN with Scala"

  setDefaultLookAndFeelDecorated(true)

  val myFont = "Monospaced"
  System.setProperty("awt.font", myFont)

  val myFontSize = 16

  // this is a light grayish color for the background
  val c537 = new Color(187, 199, 214)

  // this is a light green color for suggesting success
  val greenTint = new Color(204, 255, 179)

  // this is a light orange color for suggesting potential problems
  val orangeTint = new Color(255, 165, 0)

  // this is a light red color for suggesting a problem
  val redTint = new Color(255, 64, 0)

  // GUI spacing factors
  val borderMargin = 20
  val verticalSpaceBetweenItems = 10
  val bigVerticalSpaceBetweenItems = 30

  // popup window's inital location -- toward the top, over a ways
  var xcoord: Int = 400
  var ycoord: Int = 50

  // save these values for when we wrap around eventually
  val resetxcoord = xcoord
  val resetycoord = ycoord

  // set limits for max x and y position
  val xcoordMax: Int = 600
  val ycoordMax: Int = 600
```

```
// bump per iteration
val bumpIt: Int = 25

// End of line in Unix = \n
// End of line in Windows = \r\n
// MS Word and WordPad can cope with \n, but the default Windows text
// file editor, Notepad, fails to cope, so let's do the right thing here
val nl = System.getProperty("line.separator").toString();

// DNSDB quota values
var limit0: String = ""
var remaining0: String = ""

// the pivotal item itself, the ASN
var myasn: String = ""

// for consolidated output, we need to make sure we don't make new
// filename; we just want to use the first one we used
var FirstFileName: String = ""
var FirstDirName: String = ""

// ==================================================================
// Elements used in our GUI form

// ASN of interest
object myasn2 extends swing.TextField {
  columns = 7
  maximumSize = new Dimension(7, 24)
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}

// Max results to return per prefix
object maxresults extends swing.TextField {
  columns = 7
  maximumSize = new Dimension(5, 24)
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}

// Timefence
object gobackhowmanydays extends swing.TextField {
  columns = 5
  maximumSize = new Dimension(5, 24)
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}

// .... Radio button group
val both_v4_and_v6 = new RadioButton {
  text = "IPv4+IPv6"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
  selected = true
}
val v4_only = new RadioButton {
  text = "IPv4"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}
val v6_only = new RadioButton {
  text = "IPv6"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}
val mutex0 = new swing.ButtonGroup(both_v4_and_v6, v4_only, v6_only)
// .... Radio button group ends

// .... Radio button group
```

```
val per_prefix_output = new RadioButton {
  text = "Per-Prefix"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
  selected = true
}
val consolidated_output = new RadioButton {
  text = "Per-ASN"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}
val mutex1 = new swing.ButtonGroup(per_prefix_output, consolidated_output)
// .... Radio button group ends

// .... Radio button group
val textformat = new RadioButton {
  text = "Text      "
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
  selected = true
}
val jsonformat = new RadioButton {
  text = "JSON"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}
val mutex2 = new swing.ButtonGroup(textformat, jsonformat)
// .... Radio button group ends ....

// now do the drop down box to allow us to select the recordtype we want
val rectype = new swing.ComboBox(
  scala.List("ANY",
             "ANY-DNSSEC",
             "A",
             "AAAA",
             "NS",
             "CNAME",
             "DNAME",
             "PTR",
             "MX",
             "SRV",
             "TXT",
             "DS",
             "RRSIG",
             "NSEC",
             "DNSKEY",
             "NSEC3",
             "TLSA",
             "URI"))

// =====================================================================

// PUT UP THE BLANK GRAPHIC FORM

// When we start, make sure we have quota left...
var quota_structure = CheckRemainingQuota.check_Remaining_quota
limit0 = quota_structure._1
remaining0 = quota_structure._2

// Build the GUI interface to display
contents = new BoxPanel(Orientation.Vertical) {
  background = c537
  contents += Swing.VStrut(borderMargin)

  // Title
  contents += new BoxPanel(Orientation.Horizontal) {
    background = c537
```

```
      contents += Swing.HStrut(borderMargin)
      contents += Swing.HGlue
      contents += new swing.Label {
        text = "Checking DNSDB by ASN with Scala"
        font = new Font(myFont, java.awt.Font.BOLD, myFontSize)
      }
      contents += Swing.HGlue
      contents += Swing.HStrut(borderMargin)
    }
    contents += Swing.VStrut(bigVerticalSpaceBetweenItems)

    // Quota Report
    contents += new BoxPanel(Orientation.Horizontal) {
      background = c537
      contents += Swing.HStrut(borderMargin)
        if (limit0 == "unlimited") {
        contents += new swing.Label {
          text = "Queries/day: Unlimited"
          // this is a positive green color
          foreground = new Color(0, 102, 0)
        }
        } else if (limit0 == "DNSDB API key (in .dnsdb-apikey.txt) is missing or
invalid") {
        contents += new swing.Label {
          text = "DNSDB API key (in .dnsdb-apikey.txt) is missing or invalid"
          // this is a negative red color
          foreground = new Color(179, 0, 0)
        }
        } else {
        contents += new swing.Label {
          text = "Queries/day: " + limit0 + "    " +
            "Remaining: " + remaining0
          if (remaining0 == "0") {
            // this is a negative red color
            foreground = new Color(179, 0, 0)
          } else {
            // this is a positive green color
            foreground = new Color(0, 102, 0)
          }
      }
        }
      contents += new swing.Label {
        font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
      }
      contents += Swing.HGlue
      contents += Swing.HStrut(borderMargin)
    }
    contents += Swing.VStrut(bigVerticalSpaceBetweenItems)

    // Specify the ASN
    contents += new BoxPanel(Orientation.Horizontal) {
      background = c537
      contents += Swing.HStrut(borderMargin)
      contents += new swing.Label {
        text = "Autonomous System Number:"
        font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
      }
      contents += Swing.HStrut(5)
      contents += myasn2
      contents += Swing.HGlue
      contents += Swing.HStrut(borderMargin)
    }
    contents += Swing.VStrut(verticalSpaceBetweenItems)
```

```
// Do we want to check for IPv4 and IPv6 prefixes, or just one or the
// other?
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(15)
  contents += both_v4_and_v6
  contents += Swing.HStrut(5)
  contents += v4_only
  contents += Swing.HStrut(5)
  contents += v6_only
  contents += Swing.HStrut(borderMargin)
  contents += Swing.HGlue
}
contents += Swing.VStrut(verticalSpaceBetweenItems)

// Record Type
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(borderMargin)
  contents += new swing.Label {
    text = "Record Type:"
    font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
  }
  contents += Swing.HStrut(5)
  contents += rectype
  contents += Swing.HGlue
}
contents += Swing.VStrut(bigVerticalSpaceBetweenItems)

// Separate the optional stuff
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(borderMargin)
  contents += new swing.Label {
    text = "Optional... "
    font = new Font(myFont, java.awt.Font.ITALIC, myFontSize)
  }
  contents += Swing.HGlue
  contents += Swing.HStrut(borderMargin)
}
contents += Swing.VStrut(verticalSpaceBetweenItems)

// Max results
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(borderMargin)
  contents += new swing.Label {
    text = "Results/Prefix:"
    font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
  }
  contents += Swing.HStrut(5)
  contents += maxresults
  contents += Swing.HGlue
}
contents += Swing.VStrut(verticalSpaceBetweenItems)

// Time window
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(borderMargin)
  contents += new swing.Label {
    text = "Days Back:     "
    font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
  }
  contents += Swing.HStrut(5)
  contents += gobackhowmanydays
  contents += Swing.HGlue
}
contents += Swing.VStrut(bigVerticalSpaceBetweenItems)

// Maintain seperate files for each prefix, or consolidate into one?
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(borderMargin)
  contents += new swing.Label {
    text = "Output:"
    font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
  }
  contents += Swing.HStrut(5)
  contents += per_prefix_output
  contents += Swing.HStrut(5)
  contents += consolidated_output
  contents += Swing.HGlue
}
contents += Swing.VStrut(verticalSpaceBetweenItems)

// Output format: Text or JSON?
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(borderMargin)
  contents += new swing.Label {
    text = "Format:"
    font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
  }
  contents += Swing.HStrut(5)
  contents += textformat
  contents += Swing.HStrut(5)
  contents += jsonformat
  contents += Swing.HGlue
}
contents += Swing.VStrut(bigVerticalSpaceBetweenItems)

// search/reset/close
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(borderMargin)
  contents += swing.Button("Search") {
    get_prefixes_from_bgpview(myasn2.text)
  }
  contents += swing.Button("Reset") {
    myasn2.text = ""
    maxresults.text = ""
    gobackhowmanydays.text = ""
    mutex1.select(`per_prefix_output`)
    mutex2.select(`textformat`)
  }
  contents += swing.Button("Close") {
    sys.exit(0)
  }
  contents += Swing.HGlue
  contents += Swing.HStrut(borderMargin)
}
contents += Swing.VStrut(bigVerticalSpaceBetweenItems)

contents += new BoxPanel(Orientation.Horizontal) {
      background = c537
      contents += Swing.HStrut(borderMargin)
      contents += Swing.VStrut(10)

      contents += Swing.VStrut(verticalSpaceBetweenItems)
    } // end of vertical Frame
  }

// ====================================================================

// GET THE PREFIXES ASSOCIATED WITH THE ASN

  def get_prefixes_from_bgpview(mytempasn: String) {

    var rawJsonLines: String = ""

    // ASN should be just a number, but some people may prefix it with "as"
    // so we'll strip that if present
    var myasn = myasn2.text
    myasn = myasn.replace("AS", "")
    myasn = myasn.replace("aS", "")
    myasn = myasn.replace("As", "")
    myasn = myasn.replace("as", "")

    // build the URL for BGPview...
    val checkURL2: String = "https://api.bgpview.io/asn/" +
      myasn + "/prefixes"
    val myurl2 = new URL(checkURL2)
    val con2 = myurl2.openConnection.asInstanceOf[HttpURLConnection]

    con2.setRequestMethod("GET")
    con2.setConnectTimeout(3000)
    con2.setReadTimeout(30000)
    con2.connect()

    val ins2: InputStream = con2.getInputStream()
    val isr2: InputStreamReader = new InputStreamReader(ins2)
    val in2: BufferedReader = new BufferedReader(isr2)

    // process JSON
    rawJsonLines = in2.readLine.mkString // because this is JSON it's
    // just one longggg line..

    // we've got what we need, so close the connection
    in2.close()

    // fix the ambiguous parent prefix problem, before we parse it...
    // if we don't fix this, the parent prefixes will be included when
    // we don't want them to be
    rawJsonLines = rawJsonLines.replace("\"parent\":{\"prefix\"",
                                        "\"parent\":{\"parent_prefix\"")

    // now deserialize the prefixes; do the v4 and v6 prefixes seperately
    val elements = Json.parse(rawJsonLines)
    val my_ipv4_prefixes = (elements \ "data" \ "ipv4_prefixes" \\ "prefix")
    val my_ipv6_prefixes = (elements \ "data" \ "ipv6_prefixes" \\ "prefix")

    var myOutputFormat: String = ""
    var myPrefix: String = ""

    if (textformat.selected == true) {
      myOutputFormat = "textformat"
    } else if (jsonformat.selected == true) {
```

```
  myOutputFormat = "jsonformat"
}

// process v4 queries, if needed
if ((v4_only.selected == true) || (both_v4_and_v6.selected == true)) {
  // do each of the v4 prefixes...
  for (e <- my_ipv4_prefixes) {

    // json prefix strings include literal quotes; remove them
    myPrefix = e.toString.replace("\"", "")
    println(nl + "Prefix is = " + myPrefix)

    // now flip the slash to a comma in the prefix
    // need to fix it here for output file usage
    myPrefix = myPrefix.replace("/", ",")

    var myOutputGoesTo: String = ""
    if (per_prefix_output.selected == true) {
      myOutputGoesTo = myPrefix
    } else if (consolidated_output.selected == true) {
      myOutputGoesTo = myasn
    }

    // do the v4 query
    // example: sendQueryToDNSDB("128.223.0.0,16", "128.223.32.0.0,16")
    //          sendQueryToDNSDB("128.223.0.0,16", "3582"
    sendQueryToDNSDB(myPrefix, myOutputGoesTo)
  } // end of for (e <- my_ipv4_prefixes)
} // end of ipv4 processing block

// process v6 queries, if needed
if ((v6_only.selected == true) || (both_v4_and_v6.selected == true)) {
  for (e <- my_ipv6_prefixes) {

    // work through each of the IPv6 prefixes in this block

    // json prefix strings include literal quotes; remove them
    myPrefix = e.toString.replace("\"", "")
    println(nl + "Prefix is = " + myPrefix)

    // now flip the slash to a comma in the prefix
    // need to fix it here for output file usage
    myPrefix = myPrefix.replace("/", ",")

    // can't set where output goes until we know the relevant
    // prefix (just in case if per_prefix_output is selected)

    var myOutputGoesTo: String = ""
    if (per_prefix_output.selected == true) {
      myOutputGoesTo = myPrefix
    } else if (consolidated_output.selected == true) {
      myOutputGoesTo = myasn
    }

    // do the v6 query
    sendQueryToDNSDB(myPrefix, myOutputGoesTo)
  } // end of for (e <- my_ipv6_prefixes)
} // end of v6 processing block

// Reset the FirstFileName for next time
FirstFileName = ""
FirstDirName = ""
myasn2.text = ""
```

```
    maxresults.text = ""
    gobackhowmanydays.text = ""
    mutex1.select(`per_prefix_output`)
    mutex2.select(`textformat`)
    quota_structure = CheckRemainingQuota.check_Remaining_quota
    limit0 = quota_structure._1
    remaining0 = quota_structure._2

    println(nl + "--------------------------------")
  }

// ----------------------------------------------------------------

  def setBoxLoc() {

    xcoord += bumpIt
    if (xcoord >= xcoordMax) { xcoord = resetxcoord }

    ycoord += bumpIt
    if (ycoord >= ycoordMax) { ycoord = resetycoord }

  }

// -----------------

  def sendQueryToDNSDB(myprefix: String, OutputGoesTo: String) {

    var httpsURL: String = "https://api.dnsdb.info/lookup/rdata/ip/"

    val myasn3: String = myasn2.text

    // need a copy of this that we can modify
    val myPrefix = myprefix

    httpsURL = httpsURL + myPrefix + "/" + rectype.selection.item

    var delimiter: String = "?" // first param

    // are we limiting results returned?
    if ((maxresults.text.length > 0) &&
        (maxresults.text.toInt > 0) &&
        (maxresults.text.toInt <= 1000000)) {
      httpsURL = httpsURL + delimiter + "limit=" + maxresults.text
      delimiter = "&" // ampersand
    } else if (maxresults.text.length == 0) {
      // not specifying max results
    } else {
      val myNewWindow = new UIoutput(
        "Max Results Error...",
        "maxresults must be 1 <= maxresults <= 1000000; using default of 10000",
        orangeTint)
      setBoxLoc()
      myNewWindow.location = new Point(xcoord, ycoord)
      myNewWindow.pack()
      myNewWindow.visible = true
    }

    // are we time fencing?
    if ((gobackhowmanydays.text.length > 0) &&
        (gobackhowmanydays.text.toLong > 0) &&
        (gobackhowmanydays.text.toLong < 3650)) {
      // convert days to seconds
      val nowUnixSeconds: scala.Long = (gobackhowmanydays.text).toLong *
    (24 * 60 * 60)
  // time fence parameter is "time_last_after="
  httpsURL = httpsURL + delimiter + "time_last_after=-" + nowUnixSeconds
  delimiter = "&"
} else if (gobackhowmanydays.text.length == 0) {
  // not time fencing
} else {
  val myNewWindow = new UIoutput(
    "Time Fencing Error...",
    "maxdaysback must be 1 to 3650 integer days; doing NO time fencing",
    orangeTint)
  setBoxLoc()
  myNewWindow.location = new Point(xcoord, ycoord)
  myNewWindow.pack()
  myNewWindow.visible = true
}

val myurl = new URL(httpsURL)

// handle creating the output file
// two options exist...
// per prefix files, one per query
// per ASN files, one only, results **appended** to it for each query
//
// format 1: $HOME/dnsdb-output/$DATE/$TIME-$QUERY.txt (like normal)
// format 2: $HOME/dnsdb-output/$DATE/$TIME-$ASN.txt (time doesn't update)

// this is user's home diretory + /dnsdb-output
var homeDir = System.getProperty("user.home")
homeDir = homeDir + "/dnsdb-output"
// now add a subdirectory, y/m/d format
val dateFormat = new SimpleDateFormat("yyyy-MM-dd")
homeDir = homeDir + "/" + dateFormat.format(
  Calendar.getInstance().getTime())

// make sure a directory exists with today's date
val f: File = new File(homeDir)
f.mkdirs()

if ((myprefix == OutputGoesTo) && (FirstDirName == "")) {
  // consolidate output will go into a subdirectory time-ASnumber
  val timeFormat2 = new SimpleDateFormat("HHmmss")
  val myTime2 = timeFormat2.format(Calendar.getInstance().getTime())
  homeDir = homeDir + "/" + myTime2 + "-AS" + myasn3
  println("Making subdiredctory" + homeDir)
  val f: File = new File(homeDir)
  f.mkdirs()
  FirstDirName = homeDir
} else if ((myprefix == OutputGoesTo) && (FirstDirName != "")) {
  homeDir = FirstDirName
}

// now we need to handle making the output filename in that directory
var outputPath: String = ""
val timeFormat = new SimpleDateFormat("HHmmss")
val myTime = timeFormat.format(Calendar.getInstance().getTime())
var tempstring2: String = OutputGoesTo // OutputGoesTo is passed in...

var file_extension: String = ""
if (jsonformat.selected == true) {
  file_extension = ".jsonl"
} else if (textformat.selected == true) {
  file_extension = ".text"
```

```
} else {
  file_extension = ".???"
}

if (myprefix == OutputGoesTo) {
  // found a prefix for output, not doing consolidated output...
  // put all the prefixes in a single timestamped directory

  // fix awkwardness in potential filenames
  tempstring2 = tempstring2.replace("/", ",")
  tempstring2 = tempstring2.replace(":", "#")
  tempstring2 = tempstring2.replace("*", "STAR")
  outputPath = homeDir + "/" + myTime + "-" + tempstring2 + "-" +
    rectype.selection.item + file_extension

} else {

  // consolidated output... we want to open an ASN file, and then
  // append to it, so that all output is in one file

  if (FirstFileName == "") {
    outputPath = homeDir + "/" + myTime + "-AS" + OutputGoesTo + "-" +
      rectype.selection.item + file_extension
    FirstFileName = outputPath
  } else {
    outputPath = FirstFileName
  }
}

val f2: File = new File(outputPath)

val pw: PrintWriter = new PrintWriter(
  new FileOutputStream(f2, true /* true ==> append */ ))

val con = myurl.openConnection.asInstanceOf[HttpURLConnection]

homeDir = System.getProperty("user.home")
val apiKey = ".dnsdb-apikey.txt"
val apiPath = homeDir + "/" + apiKey

val filename = apiPath
var myLine = ""
for (line <- Source.fromFile(filename).getLines()) {
  myLine = line
}

con.setRequestProperty("X-API-Key", myLine)
con.setRequestProperty("User-Agent", "DNSDB Check-By-ASN Demo Client")

if (mutex2.selected.get == `jsonformat`) {
  con.setRequestProperty("Accept", "application/json")
} else if (mutex2.selected.get == `textformat`) {
  con.setRequestProperty("Accept", "text/plain")
}

con.setRequestMethod("GET")
con.setConnectTimeout(3000)
con.setReadTimeout(90000)

var responseCode = 9999 // assume we couldn't do the query by default

scala.concurrent.blocking {
  con.connect()
```

```
responseCode = con.getResponseCode()

if (responseCode == 404) {
  // no records found
} else if (responseCode == 400) {
  val myNewWindow =
    new UIoutput("Query Error...", "URL formatted incorectly", redTint)
  setBoxLoc()
  myNewWindow.location = new Point(xcoord, ycoord)
  myNewWindow.pack()
  myNewWindow.visible = true
} else if (responseCode == 403) {
  val myNewWindow = new UIoutput("Query Error...",
                                 "X-API-Key header not present or wrong",
                                 redTint)
  setBoxLoc()
  myNewWindow.location = new Point(xcoord, ycoord)
  myNewWindow.pack()
  myNewWindow.visible = true
} else if (responseCode == 429) {
  val myNewWindow =
    new UIoutput("Query Error...",
                 "API daily key quota exceeded",
                 redTint)
  setBoxLoc()
  myNewWindow.location = new Point(xcoord, ycoord)
  myNewWindow.pack()
  myNewWindow.visible = true
} else if (responseCode == 429) {} else if (responseCode == 500) {
  println("Error processing request" + nl)
} else if (responseCode == 503) {
  val myNewWindow =
    new UIoutput("Query Error...",
                 "Too many concurrent queries",
                 redTint)
  setBoxLoc()
  myNewWindow.location = new Point(xcoord, ycoord)
  myNewWindow.pack()
  myNewWindow.visible = true

} else if (responseCode == 200) {
  println("Output is going to = " + outputPath)
  /* val myNewWindow =
  new UIoutput(
    "Output is being sent to... (please be patient for large queries!)",
    outputPath,
    greenTint)
setBoxLoc()
myNewWindow.location = new Point(xcoord, ycoord)
myNewWindow.pack()
myNewWindow.visible = true
   */

  val ins: InputStream = con.getInputStream()
  val isr: InputStreamReader = new InputStreamReader(ins)
  val in: BufferedReader = new BufferedReader(isr)
  var ok = true

  while (ok) {
    val lines = in.readLine.mkString
    ok = lines != null
    if (ok) {
      pw.write(lines + nl)
```

```
          }
        }

        pw.write(nl)

        in.close()
        pw.close()

      }
    }

    // update available quota
    val quota_structure = CheckRemainingQuota.check_Remaining_quota
    limit0 = quota_structure._1
    remaining0 = quota_structure._2
  } // end of UI
}

object CheckASN {

  def main(args: Array[String]) = {

    if (CheckAPI.CheckAPIFile.toString == "OK") {

      // all's okay, proceed as normally
      val ui = new UI
      ui.location = new Point(50, 50)
      ui.pack()
      ui.visible = true

      while (System.in.read() != -1) {}
    } // APIFileIsOK block
  } // main
} // CheckASN

// -----------------

class UIoutput(frameTitle: String, fileLocation: String, MyBoxColor: Color)
    extends swing.Frame {

  // Create a new wide and short window for posting one-line log-like info
  // (can be closed without affecting the main window)

  preferredSize = new Dimension(760, 60)
  setDefaultLookAndFeelDecorated(true)

  title = frameTitle // passed in as a parameter

  contents = new BoxPanel(Orientation.Horizontal) {
    // Adding the BoxPanel lets me left-justify the label
    background = MyBoxColor // passed in as a parameter
    contents += Swing.HStrut(10) // preserve a left margin

    contents += new swing.Label {
      text = fileLocation // passed in as a parameter
      val myFont = "Monospaced"
      val myFontSize = 16
      font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
    }

    contents += Swing.HGlue // left justify
    contents += Swing.HStrut(10) // preserve a right margin
  }}
```

Appendix 4. CheckASN/src/main/scala/CheckAPI.scala

```scala
package checkasn

import java.io._
import java.lang._
import scala._
import scala.io._

// ----------------

object CheckAPI {

  def CheckAPIFile: String = {

    // Does the API Key File exist, and is it of at least the right size?

    val homeDir0 = System.getProperty("user.home")
    val apiKey0 = ".dnsdb-apikey.txt"
    val apiPath0 = homeDir0 + "/" + apiKey0

    val f = new File(apiPath0)
    val filename0 = apiPath0
    var myLine0 = ""

    if (f.exists()) {
      for (line0 <- Source.fromFile(filename0).getLines()) {
        myLine0 = line0
        myLine0 = myLine0.replaceAll("(\\r|\\n)", "")
      }
    } else {
      val uiError2 = new UIerror2
      uiError2.visible = true
      // we kill the main window after showing the error dialog
    }

    val fileContentsLength = myLine0.length()

    if (fileContentsLength == 64) {
      // file contents length is correct
    } else {
      // println("Wrong size : " + fileContentsLength)
      val uiError = new UIerror
      uiError.visible = true
      // we kill the main window after showing the error dialog
    }

    // made it this far? all's superficially okay with the API key file,
    // so proceed as normal
    return "OK"
  }
}
```

Appendix 5. CheckASN/src/main/scala/CheckRemainingQuota.scala

```
package checkasn

import java.io._
import java.lang._
import java.net._
import scala.io._

object CheckRemainingQuota {

  def check_Remaining_quota:(String, String) = {

    var limit0: String = ""
    var remaining0: String = ""

    // End of line in Unix = \n
    // End of line in Windows = \r\n
    // MS Word and WordPad can cope with \n, but the default Windows text
    // file editor, Notepad, fails to cope, so let's do the right thing here
    val nl = System.getProperty("line.separator").toString();

    val homeDir0 = System.getProperty("user.home")
    val apiKey0 = ".dnsdb-apikey.txt"
    val apiPath0 = homeDir0 + "/" + apiKey0

    val filename0 = apiPath0
    var myLine0 = ""
    for (line0 <- Source.fromFile(filename0).getLines()) {
      myLine0 = line0
    }

    val checkURL0: String = "https://api.dnsdb.info/lookup/rate_limit"
    val myurl0 = new URL(checkURL0)
    val con0 = myurl0.openConnection.asInstanceOf[HttpURLConnection]

    con0.setRequestProperty("X-API-Key", myLine0)
    con0.setRequestProperty("User-Agent", "DNSDB Check-By-ASN Demo Client")
    con0.setRequestProperty("Accept", "text/plain")
    con0.setRequestMethod("GET")
    con0.setConnectTimeout(3000)
    con0.setReadTimeout(3000)
    con0.connect()
    val MyresponseCode0 = con0.getResponseCode()

    if (MyresponseCode0 == 404) {
      // println("Quota check URL not found" + nl)
    } else if (MyresponseCode0 == 400) {
      // println("Quota check URL formatted incorrectly" + nl)
    } else if (MyresponseCode0 == 403) {
      // println("X-API-Key header not present or wrong" + nl)
      limit0 = "DNSDB API key (in .dnsdb-apikey.txt) is missing or invalid"
    } else if (MyresponseCode0 == 500) {
      println("Error processing quota check request" + nl)
    } else if (MyresponseCode0 == 200) {
      val ins0: InputStream = con0.getInputStream()
      val isr0: InputStreamReader = new InputStreamReader(ins0)
      val in0: BufferedReader = new BufferedReader(isr0)

      // ignore three lines
      var lines0 = in0.readLine.mkString
      lines0 = in0.readLine.mkString
      lines0 = in0.readLine.mkString
      // fourth line = limit (we want this one)
      lines0 = in0.readLine.mkString
      lines0 = lines0.replace("\"limit\":","")
      lines0 = lines0.replace(" ","")
      lines0 = lines0.replace(",","")
      limit0 = lines0


      // fifth line = remaining (we want that one, too)
      lines0 = in0.readLine.mkString
      lines0 = lines0.replace("\"remaining\":","")
      lines0 = lines0.replace(" ","")
      remaining0 = lines0

      // dump six and seventh lines
      lines0 = in0.readLine.mkString
      lines0 = in0.readLine.mkString

      in0.close()

    }
    return (limit0, remaining0)
  }
}
```

Appendix 6. CheckASN/src/main/scala/UIerror.scala

```
package checkasn

import scala._
import scala.swing._
import swing._

class UIerror extends MainFrame {
  contents = new BoxPanel(Orientation.Vertical) {
    contents += new Label {
      text = " "
    }
  }
  val res = Dialog.showConfirmation(
    contents.head,
    ".dnsdb-apikey.txt file in the default home directory should be\nexactly 64
characters long (disregarding end-of-line characters).\n\nIT'S NOT. Cannot
continue.\n\nHit either button to exit. Fix the file, then try again.",
    optionType = Dialog.Options.OkCancel,
    title = "ERROR!"
  )
  if (res == Dialog.Result.Ok)
    sys.exit(0)
  else
    sys.exit(0)
} // end of UIerror
```

Appendix 7. CheckASN/src/main/scala/UIerror2.scala

```
package checkasn

import scala._
import scala.swing._
import swing._

class UIerror2 extends MainFrame {
  contents = new BoxPanel(Orientation.Vertical) {
    contents += new Label {
      text = " "
    }
  }
  val res = Dialog.showConfirmation(
    contents.head,
    "Your DNSDB API key MUST be in a plain text file called .dnsdb-apikey.txt in the
default home directory.\n\nWE CAN'T FIND IT.\n\nDid you forget to create it?\n\nOr
perhaps you misnamed it by forgetting the leading dot in the file name? \n\nMaybe it's in
the wrong directory?\n\nAnyhow, without it, WE CANNOT CONTINUE.\n\nHit either button to
exit. Get the file in the right spot, then try again.",
    optionType = Dialog.Options.OkCancel,
    title = "ERROR!"
  )
  if (res == Dialog.Result.Ok)
    sys.exit(0)
  else
    sys.exit(0)
} // end of UIerror2
```

Appendix 8. CheckASN/build.sbt

```
name := "CheckASN"

version := "1.0"

scalaVersion := "2.12.2"

cancelable in Global := true

fork in run := true

connectInput in run := true

enablePlugins(JavaAppPackaging)

enablePlugins(WindowsPlugin)

maintainer := "Name <email@somedomain here>"
packageSummary := "QueryDNSbyASN"
packageDescription := """Demonstrate querying DNSDB by ASN with Scala"""

// Get GUIDs from https://www.guidgen.com/
wixProductId := "add GUID here"
wixProductUpgradeId := "add another GUID here"

// javaOptions in run += "-Djavax.net.debug=ssl:handshake"

// https://mvnrepository.com/artifact/org.scala-lang.modules/scala-swing_2.12
libraryDependencies += "org.scala-lang.modules" % "scala-swing_2.12" % "2.0.0"

// https://mvnrepository.com/artifact/com.typesafe.play/play-json_2.12
libraryDependencies += "com.typesafe.play" % "play-json_2.12" % "2.6.0-M6"
```