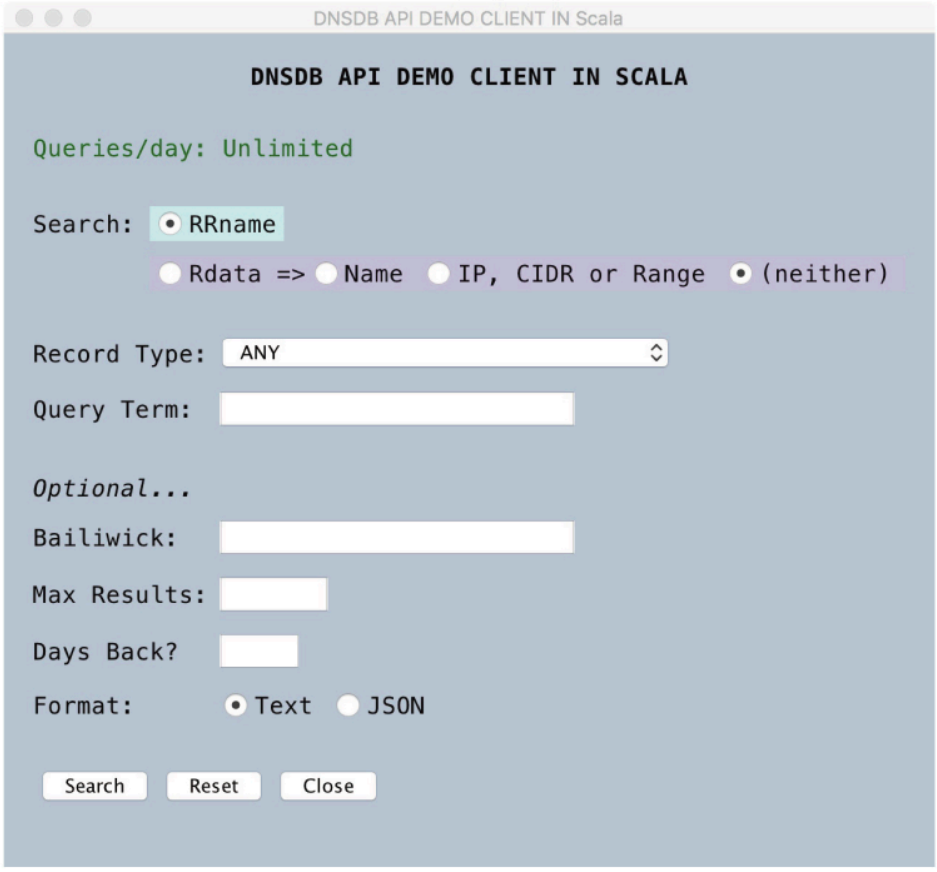
FARSIGHT SECURITY

White Paper

Building A Demo GUI Front End For DNSDB API In Scala With Swing For The Mac And For Windows PCs

v1.0.1, April 28, 2017

Joe St Sauver, Ph.D. (stsauver@fsi.io)



Preface

This white paper will explain how to use Scala with Swing to create a demonstration GUI interface to Farsight's passive DNS database, DNSDB API.

Before getting into the meat of that, let's go over some prefatory information.

Intended Audience

This white paper is meant for users comfortable with detailed technical content. Experience with some sort of programming or scripting language will be helpful, and help you to get more out of this document.

Substantively, the functionality described in this white paper may also be of particular interest to certain DNSDB users, including users who work in a pure Microsoft Windows PC environment.

Organization of This White Paper

This is a relatively long white paper. If you review the table of contents, that will allow you to see how this paper is organized, and what we'll cover. In looking at table of contents, don't let the number of "chunks" deter you -- some "chunks" may consist of just a paragraph or two of text. There are also lots of images and other figures.

Another thing we've done to try to reduce the paper's complexity a bit is to divide it into three main sections plus five appendicies:

1) The first section consists of some introductory narrative/discussion (which you can skip if you're impatient, although reading this material will help you to understand "where we're coming from" and why we did some of the things we did)

2) A second section, showing you how to actually build the demo GUI front end code for use on a local Mac, and

3) A third section, focusing on packaging and porting the code for use on third party systems, including both 3rd-party Macs and 3rd-party Microsoft Windows PCs (such as systems running Windows 10 or Windows 7).

The appendicies contain copies of our actual code.

Table of Contents

Preface 2
Table of Contents 3
Section 1. Discussion/Narative
I. Introduction 4
II. Why Scala 4
III. Scala Documentation 5
IV. Our Overall Plan For Today's Example 6
V. Integrated Development Environment? 7
VI. Graphical User Interface Selection 8
VII. GUI Layout Managers 8
VIII. RESTFUL API Call Format 11
IX. Cryptographic Scaffolding 12
X. Creating build.sbt sbt Config Files 13
Section 2. Actually Building/Installing Code On The Local Machine
XI. Getting Started With Scala On The Mac: Prerequisites 16
XII. Installing Scala and sbt 16
XIII. Creating The Directory Structure We Need 17
XIV. SampleScala/build.sbt 18
XV. SampleScala/project/assembly.sbt 18
XVI. SampleScala/project/plugins.sbt 19
XVII. Our Actual Scala Source Code 19
XVIII. Creating Your DNSDB API Key File 19
XIX. The JCE Unlimited Strength Jurisdiction Policy Files 21
XX. Building The Scala Code 25
XXI. Running A Query and Looking At The Results 26
XXII. Quitting 28
XXIII. Packaging Our Code For Routine Local Use Without sbt 29
Section 3. Packaging For Use on 3rd-Party Macs and Windows PCs
XXIV. Creating A dmg Package For Use on A 3rd-Party Mac 30
XXV. Running Our dmg On A 3rd-Party Mac 31
XXVI. Making A Microsoft Windows Installer 32
XXVII. Using The MS Windows Installer On A 3rd-Party System 35
Section 4. Conclusion
XXVIII. Miscellaneous Notes 42
XXIX. What Didn't We Do? 44
XXX. Conclusion 45
XXXI. Acknowledgments 45
Appendicies
Appendix I. SampleScala/build.sbt file 46
Appendix II. SampleScala/src/main/scala/SampleScala.scala 47
Appendix III. SampleScala/project/assembly.sbt 63
Appendix IV. SampleScala/project/plugins.sbt 63
Appendix V. Licenses 64
Appendix VI. check.java source file 65

SECTION 1. Discussion/Narrative

I. Introduction

Many of Farsight's DNSDB customers access DNSDB via either:

- Our command line interface API demo programs **dnsdb_query.py**[1], and/or its C language analog, **dnsdb_query**,[2] or via

- Our **point-and-click web interface**[3]

However, because DNSDB uses a RESTful API, there's no reason why you shouldn't be able to access DNSDB from virtually any programming language.

For any "old school"/traditional coders out there, we've previously demonstrated how to query DNSDB using libcurl from gcc**,** see *"Making Programmatic DNSDB Queries With libcurl."*[4]

This document describes building a GUI interface to DNSDB using Scala on the Mac.

We'll also package our program so we can move it over and run it on other Macs, and on common versions of Microsoft Windows (such as Microsoft Windows 7 or Microsoft Windows 10).

II. Why Scala?

You may wonder, why use Scala rather than directly use the world's 2nd-most-popular programming language, **Java**?[5]

Answer: Scala fixes many of the frustrations that have historically bedeviled Java, while still giving coders the ability to use Java's capabilities when doing so is convenient (or inescapable).

To make that concrete, see (the somewhat tongue-in-cheek) *"How Scala compares with 20 other programming languages according to Reddit analysis:"*[6]

> *[...] Scala appears to be the only production-proven programming language to make engineers happy and alleviate their need to curse [relative to] more broadly-adopted languages like Java, PHP and JavaScript.*

That analysis, while couched a bit wryly, is based in truth. Scala is "fun to code in" in a way that Java and other more-pedantic/strictly-object-oriented languages tend not to be. Less cussing and more happiness results.

---

[1] https://github.com/dnsdb/dnsdb-query
[2] https://github.com/dnsdb/dnsdb_c/
[3] https://www.dnsdb.info/
[4] https://www.farsightsecurity.com/2016/11/04/stsauver-dnsdb-libcurl/
[5] http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages
[6] https://www.lightbend.com/blog/how-scala-compares-20-programming-languages-reddit-analysis

Scala also naturally supports a functional programming model. Quoting Alvin Alexander's online functional programming book:[7]

> *As at least one functional developer has said, when you , "Think in FP," you see an application as (a) data flowing into the application, (b) being transformed by a series of transformer functions, and (c) producing an output. The first lessons in this book are aimed at helping you to "Think in FP," to see your applications in this way, as a series of data flows and transformers.*

In my opinion, that's an excellent way to think in general when working with Farsight's Security Information Exchange, and Farsight's passive DNS database, DNSDB.

Finally, if none of the above convinces you to become interested in Scala, perhaps you're "monetarily-motivated." If so, note that Stack Overflow's annual survey of *"Top Paying Technologies by Region"*[8] lists Scala as being tied (with Golang) as the highest-paid programming language in the United States at $110,000/year. The market rewards high-capability/high-demand/low-availability skillsets with appropriate salaries!

III. Scala Documentation

Excellent online documentation for Scala is available at the Scala-Lang.Org **documentation web site,**[9] including a good **overview document**.[10]

There are many other excellent online Scala resources, too -- too many to list them all here. Nonetheless, if you've not worked with Scala before, I'd strongly urge you to check out:

1. Otfried Cheong's **Scala pages**[11]

2. Alvin Alexander's great **online Scala materials,**[12] as referenced throughout this article. Alvin also authored the magnificent *"Scala Cookbook"* for O'Reilly (see below) and has an online book, *"Functional Programming, Simplified"* (also mentioned previously in this post).

3. Odersky, Spoon, and Venners's **"Programming in Scala," 1st edition,**[13] freely available online. Rather have the current (3rd) edition? See below for a link to that traditionally published book.

Speaking of traditional-format books, a variety of traditionally-published Scala books are also available in print (or e-book) form from Amazon or your favorite local independent technical bookstore. See for example (in alphabetical order by author's last name):

---

[7] http://scalafp.com/book/goals.html
[8] http://stackoverflow.com/insights/survey/2017/
[9] https://www.scala-lang.org/documentation/
[10] https://www.scala-lang.org/docu/files/ScalaOverview.pdf
[11] http://otfried.org/scala/
[12] http://alvinalexander.com/scala
[13] http://www.artima.com/pins1ed/

1. Alvin Alexander's **"Scala Cookbook."**[14]

2. Cay S. Horstmann's **"Scala for the Impatient"**[15]

3. Odersky, Spoon and Venner's **"Programming in Scala: Updated for Scala 2.12"**[16]

4. Nilanjan Raychaudhuri's **"Scala in Action"**[17]

5. Venkat Subramaniam's **"Pragmatic Scala: Create Expressive, Concise, and Scalable Applications"**[18]

6. Joshua D. Sureth's **"Scala in Depth"**[19]

7. Jason Swartz's **"Learning Scala: Practical Functional Programming for the JVM"**[20]

8. Wampler and Payne's **"Programming Scala: Scalability = Functional Programming + Objects"**[21]

There are doubtless other terrific Scala books, too, sorry if we accidentally omitted any which you may particularly like.

IV. Our Overall Plan For This Example

In order for this project to work, we need to be able to:

1. Display a form in a graphical user interface (GUI) window. We'll roughly model our interface on Farsight's classic web interface to DNSDB, but just for the heck of it, let's extend the capabilities our interface offers to include the ability to:

    a. See how many queries we've consumed

    b. Specify the maximum number of observations we want to potentially receive, from one to one million.

    c. Limit the returned results to those from just the last *N* days, and

    d. Get those results in either plain text or **JSON lines**[22] formats.

---

[14] https://www.amazon.com/gp/product/1449339611/
[15] https://www.amazon.com/Scala-Impatient-Cay-S-Horstmann/dp/0321774094/
[16] https://www.amazon.com/Programming-Scala-Updated-2-12/dp/0981531687/
[17] https://www.amazon.com/Scala-Action-Covers-2-10/dp/1935182757/
[18] https://www.amazon.com/Pragmatic-Scala-Expressive-Scalable-Applications/dp/1680500546/
[19] https://www.amazon.com/Scala-Depth-Joshua-D-Suereth/dp/1935182706/
[20] https://www.amazon.com/Learning-Scala-Practical-Functional-Programming/dp/1449367933/
[21] https://www.amazon.com/Programming-Scala-Scalability-Functional-Objects/dp/1491949856/
[22] http://jsonlines.org/

2. Take the information the user entered into that form and use it to do a **RESTful**[23] query against DNSDB API over a strongly-encrypted TLS connection (including handling authentication via DNSDB's API key header). Because we're highly interested in promoting adoption of strong cryptography, we've intentionally used a ciphersuite that will give you few options but to download and install Java's unlimited strength crypto policy files.

3. We also need to manage the results returned by DNSDB. For this program, we'll save those results to text files, one per query, and leave viewing and other management of those files to the system's file manager (e.g., Finder in the case of the Mac or Windows Explorer in the case of PCs running Microsoft Windows). It might be tempting to display our output in a graphic window, but recognize that a single query may return up to a *million* results. That's hard to handle elegantly in a GUI window. To keep this already-long white paper semi-straightforward, we're just going to punt and write files. (We will, however, at least give the user a "popup" window explaining where to look for those files)

IMPORTANT NOTE: The client described in this white paper is meant just as an ***illustration/example,*** it is **NOT** being released as an officially-supported additional production interface to Farsight's DNSDB API. Remember -- the code you're seeing here is JUST meant as an example/illustration, not production/bulletproofed code!

V. Integrated Development Environment?

Scala code can be built with a regular text editor, or using an integrated development environment (such as **NetBeans**,[24] **Eclipse**,[25] or **IntelliJ**[26]). We won't be installing or using an IDE for the purposes of this article.

If you independently decide to try experimenting with a Scala-aware IDE, please note that at least in some cases, it may be difficult or impossible to fully rollback such installations should you want to do so. For example, the **Eclipse FAQ**[27] states that

> *Depending upon the particular version of Eclipse you are running with, the difficulty of uninstalling features or plugins from an Eclipse installation ranges from trivial to painful. In some cases, Eclipse doesn't support uninstalling certain 'optional' features after they have been installed.*

Proceed carefully if you decide to install an IDE notwithstanding that caveat, and be sure you have a current backup.

---

[23] https://en.wikipedia.org/wiki/Representational_state_transfer
[24] https://netbeans.org/
[25] https://eclipse.org/
[26] https://www.jetbrains.org/
[27] https://wiki.eclipse.org/FAQ_How_do_I_remove_a_plug-in%3F

VI. Graphical User Interface Selection

For our GUI, we'll use the Scala implementation of Swing. Some online Scala Swing resources you should keep in mind include:

1. The **scala-swing**[28] site on Github

2. The **scala-swing incubation project**[29] site on Github, and

3. The **Scala Swing 2.12 libraries**[30] on the Maven Repository.

Scala does support other GUIs, including **JavaFX,**[31] but Swing's good enough for our little example.

Documentation for Scala Swing is relatively scarce, but check out classic Java Swing books and the Swing section of the **Java Client Technologies website.**[32] You may also want to see the **Scala-Swing project site**[33] on Github.

VII. GUI Layout Managers

There's also the matter of choice of layout managers.

Layout managers help applications figure out a graceful way to reconfigure a user's interface when confronted with different screen geometries (such as huge LCD panels vs small mobile devices with tiny screen geometries), different font sizes, etc. Layout managers usually accommodate those odd screen and font sizes by moving around or resizing buttons, fields, labels, panels, etc., all while attempting to preserve the relationship between items and an overall-usable/reasonable display configuration.

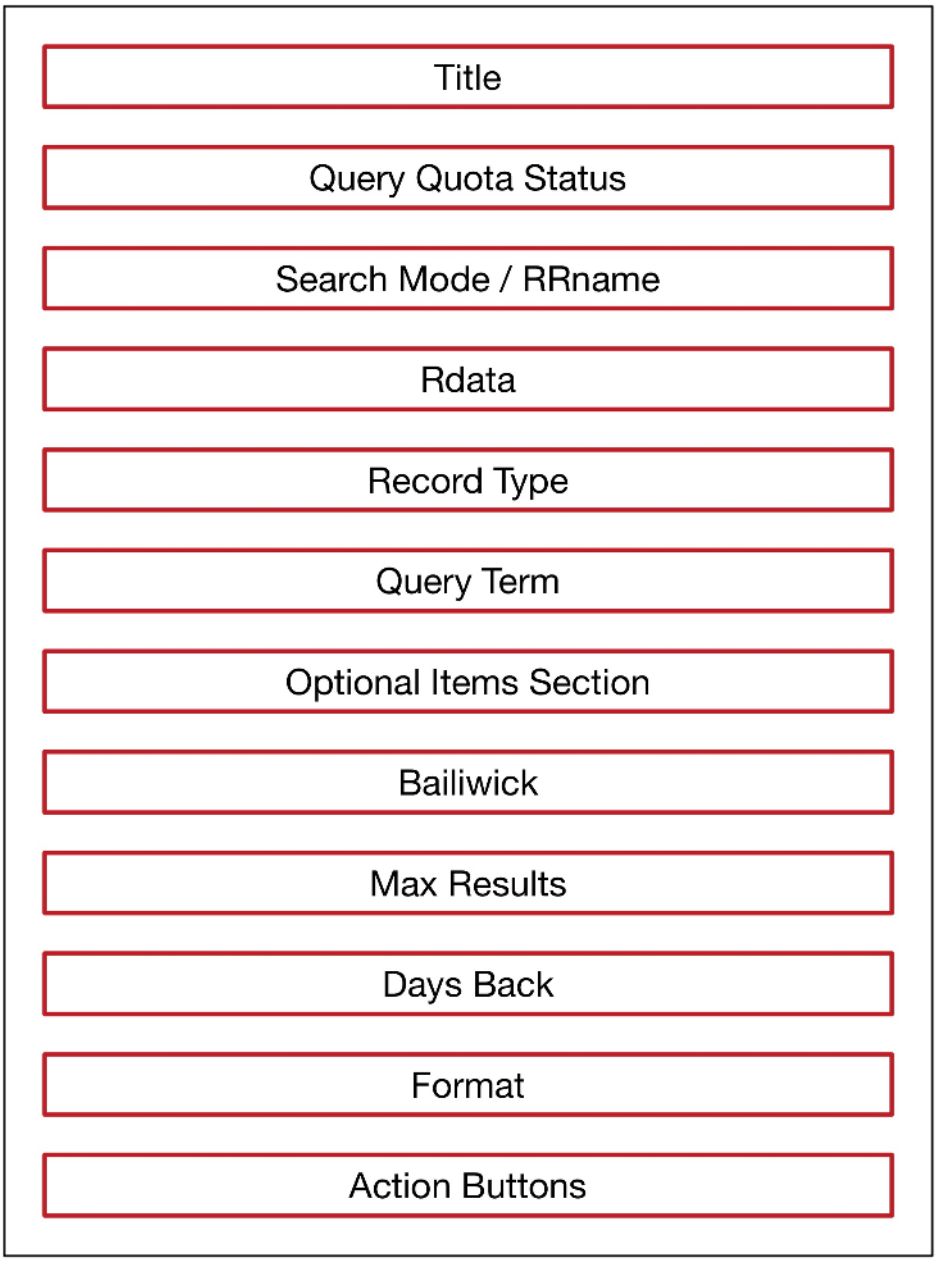
Since this is just a demo client, we're simply going to hard code the interface layout. We've made it the way we want it to look on a typical laptop screen rather than bothering to use a layout manager. (The tradeoff is obviously one of control and simplicity vs. flexibility and elegance).

The overall architecture of the program is very straightforward. We created a vertical *BoxPanel* and then a series of horizontal *BoxPanels* within that vertical *BoxPanel*, one for each field. See Figure 1.

---

[28] https://github.com/scala/scala-swing
[29] https://github.com/ingoem/scala-swing
[30] https://mvnrepository.com/artifact/org.scala-lang.modules/scala-swing_2.12
[31] http://docs.oracle.com/javase/8/javase-clienttechnologies.htm
[32] *ibid*
[33] https://github.com/scala/scala-swing/blob/2.0.x/docs/SIP-8.md

Figure 1. GUI Panel Conceptual Layout

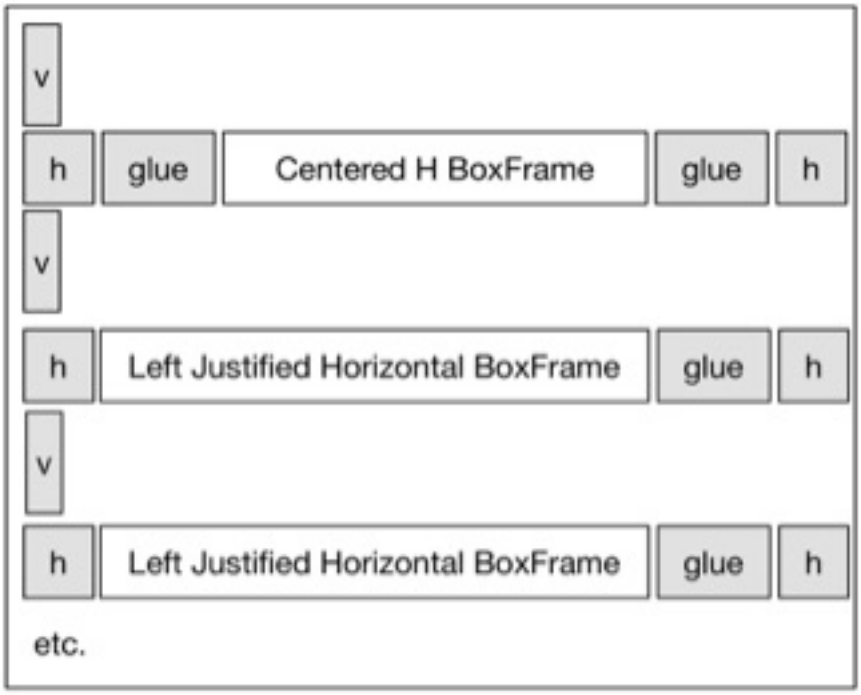


Boxes were spaced using Scala Swing's *strut* and *glue* features. *struts* and *glue* will feel very familiar to anyone who's ever worked laying out documents (particularly tables!) with TeX or LaTeX.

struts (see the shaded "v" and "h" boxes below in figure 2) act as rigid horizontal or vertical spaces. We use them today to maintain margins, and for inter-item spacing.

glue (actually, a somewhat more descriptive name might be "expanding foam") automatically expands to fill available space, with glue on the right translating to a left-justified layout, and glue on the left *and* right simultaneously creating centered text. See Figure 2.

Figure 2. Use of Horizontal/Vertical Struts and Glue For Spacing



The layout components for the topmost box in that diagram can be seen in the following code excerpt.

Figure 3. Scala Swing Code Excerpt Illustrating Use of Horizontal/Vertical Struts and Glue For Spacing

```
def do_GUI_form() {
  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 = "DNSDB API DEMO CLIENT IN SCALA"
        font = new Font(myFont, java.awt.Font.BOLD, myFontSize)
      }
      contents += Swing.HGlue
      contents += Swing.HStrut(borderMargin)
    }
    contents += Swing.VStrut(bigVerticalSpaceBetweenItems) [...]
```

Spacing considerations also helped to drive our decision to use Scala's *Monospaced* font. While we could have created a table layout, or established tab settings, our decision to use a monospace font simplified the process of visually aligning horizontal fields since all characters are of equal width.

We also made the "executive decision" to use relatively large fonts, as you might if you have a large display visual accuity challenges. Since you have source code, if you like a different size font, you obviously can change the source and rebuild.

VIII. RESTFUL API Call Format

Once we've gleaned the information we need from our form, we then need to build and execute RESTFUL API calls to the DNSDB API.

The specifications for the DNSDB API are available **online,**[34] so it's just a matter of translating what had been collected in our online form into what the DNSDB API actually needs, and then making a RESTFUL API call for the query.

Making a RESTFUL API call is generally a straightforward matter of assembling the required parameters into a complete URL, although some "conversions" may be required for some user-specified parameters.

For instance, we need to convert "days" (as used in the interface for time fencing purposes) into seconds (as required by DNSDB API). While date and time values always have the potential to present unexpected complexities and there are specific authoritative routines for doing time manipulations, in this case we simply elected to convert days into seconds by multiplying the number of days by (24 hour/day) * (60 minutes/hour) * (60 seconds/minute).

We also need to handle characters that can't be passed without risk of being misinterpreted, as in the case of slash characters -- is a slash character part of a CIDR mask? Or does a slash character represent a separator between directories and file names? The DNSDB API specifies that slashes (as used in the representation of CIDR prefixes such as *128.223.0.0/16*) must be replaced (for DNSDB API purposes) with commas (resulting in CIDR specifications such as *128.223.0.0,16*).

This and a couple of other tweaks can easily be made with the Scala *replace* statement. For example, if you look at the Scala code in Appendix II, you'll see the statements shown in Figure 4:

Figure 4. Scala replace statements

```
tempstring2 = tempstring2.replace("/", ",")
tempstring2 = tempstring2.replace(":", "#")
tempstring2 = tempstring2.replace("*", "STAR")
```

Note that you don't need to make a one-character-for-one character substitution. This can be seen in the last of those statements, where we replace an asterisk (one character) with a four character-long string literal ("STAR").

[34] https://api.dnsdb.info/

IX. Cryptographic Scaffolding

Farsight's public API (like all sane sites that authenticate with non-one-time credentials) uses an *https* (secure) web interface. This means that any client that wants to connect to the DNSDB API MUST be able to establish an SSL/TLS (secure) connection. But what cryptographic framework/toolkit should we use for that?

Perhaps libcurl? In our previously-mentioned **blog post**,[35] we used **libcurl**[36] on top of **OpenSSL**[37] as our higher-level encrypted data transport environment. That worked well.

Unfortunately, there are no **libcurl Scala bindings.**[38] There ARE **Java bindings for libcurl,**[39] but that's reportedly an incomplete/in-process effort.

To avoid having the "simple" matter of our choice of crypto scaffolding devolve into a long review of Java cryptographic options and their respective strengths and shortcomings, we'll just be pragmatic and use the **Java Net URL Class**[40] with the **Java HttpURLConnection Class.**[41]

Figure 5.

```
openConnection.asInstanceOf[HttpURLConnection]
```

That approach is not without its limitations, but it's good enough for our purposes. The basics of that approach (albeit in Java rather than Scala) are well illustrated in yet another excellent article by **Alvin Alexander.**[42]

An aside: **Dispatch**[43] is often mentioned in online fora as an alternative worth considering for https connections in Scala, but Dispatch is known for using "cryptic operator methods" that can make your program read like "Morse Code" -- see for example **these comments.**[44] If you do decide to try rewriting our sample code to use a different https library, such as Dispatch, you may find the rather cool **"Periodic Table of Dispatch Operators"**[45] to be a helpful online "cheatsheet."

In thinking further about issuing our restful calls, note that because the DNSDB API uses an API key for authentication, we also need to be able to send our API key as a supplemental header. (This is not typically a big deal, just one of those things we need to know how to accommodate).

---

[35] "Making Programmatic DNSDB Queries With libcurl," https://www.farsightsecurity.com/2016/11/04/stsauver-dnsdb-libcurl/
[36] https://curl.haxx.se/libcurl/
[37] https://www.openssl.org/
[38] https://curl.haxx.se/libcurl/bindings.html
[39] https://curl.haxx.se/libcurl/bindings.html
[40] https://docs.oracle.com/javase/8/docs/api/java/net/URL.html#openConnection--
[41] https://docs.oracle.com/javase/8/docs/api/java/net/HttpURLConnection.html
[42] http://alvinalexander.com/blog/post/java/simple-https-example
[43] http://dispatch.databinder.net/Dispatch.html
[44] http://stackoverflow.com/questions/5564074/scala-http-operations
[45] http://www.flotsam.nl/dispatch-periodic-table.html

Headers will also be how we choose between plain text and JSON lines format output. (Speaking of, for this example, we're going to save all our output with a .txt file extension, even if the JSON lines format output could arguably be more accurately served with a .jsonl extension)

We also want to be able to explicitly "fine tune" the cryptography we use so as to "encourage" users to install Java's unlimited strength crypto policy files.

We do that by forcing the use of a particular cipher suite, namely what the IETF calls *TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384* (that ciphersuite is also known by its "OpenSSL name," *ECDHE-RSA-AES256-GCM-SHA384*). This is a strong cipher suite, and one of the ciphersuites that's part of the **"Better Crypto Applied Crypto Hardening"**[46] "Strong Cipher Suites Configuration A" (and B) recommendations. So far, so good.

The bad part? That cipher suite will **ONLY** work if we install Java's Unlimited Strength Cryptography Policy Files, as we describe later in this article. If you:

- Don't trust elliptic curve crypto, or
- You'd prefer to just let regular ciphersuite negotiation occur, or
- You're not in a locale where access to strong crypto is a legal option,

you can just comment out the lines selecting that cipher suite in the provided Scala source code, see Appendix II.

A final cryptography-related point: some documents suggest that 2048 bit Ephemeral Diffie Hellman is the maximum potentially available DHE key size in Java 1.8 (see for example "Clients support Ephemeral DH over 1024 bits" BUT "Somewhat less positively, the new limit is 2048 bits" as mentioned in Ivan Ristic's article **"Significant SSL/TLS improvements in Java 8."**[47] In our hands-on testing, however, we've found that 4096 bit EDH CAN BE successfully used, so that's what we've configured in our sample code. Again, if you don't want this or can't use this do to policy constraints, this option can be removed from the sample code.

X. Creating *build.sbt* sbt Config Files

Scala and *sbt* are continually-evolving languages. The most recent version of *sbt* now supports multiproject builds. The recommended *build.sbt* format has changed as a result. See the **sbt build definition page**[48] for details.

Because of the relative simplicity of this project, however, we decided to just create a simple "classic-format" *build.sbt* file per the *sbt* Reference Manual's **Appendix: .scala build definition,**[49] instead. My impression is that the classic format is somewhat simpler than the new format, and the classic format continues to work just fine.

[46] https://bettercrypto.org/static/applied-crypto-hardening.pdf
[47] https://blog.ivanristic.com/2014/03/ssl-tls-improvements-in-java-8.html
[48] http://www.scala-sbt.org/0.13/docs/Basic-Def.html
[49] http://www.scala-sbt.org/0.13/docs/Full-Def.html

Working with *build.sbt* also brings up the question of dependency management, as in "How do we find any additional libraries we may need for our code?"

Java allows an "unmanaged dependency" model where you simply dump the jar files you're using into your Java **classpath.**[50] Scala and sbt will then find and use the files they need. While that *laissez faire* approach works OK, we prefer to use the less-*ad-hoc*/more-structured "managed dependencies" model. In the managed dependencies model, you'll need to explicitly tell sbt where to download the Scala libraries you want to use.

In our *build.sbt* file, as shown in Appendix I, there's only one such dependency -- it's the scala-swing library. That dependency is specified as:

Figure 6. Sample build.sbt dependency expression

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

Where did those *specific* lines come from? Well, we copied and pasted them from the sbt "tab" on the **scala-swing**[51] page at the Maven Repository.

Speaking of Maven, when searching for artifacts in the Maven Repository, it is important that you get the *RIGHT VERSION* of the libraries you need. If you just search Maven for scala-swing , the first "scala-swing" version Maven finds (at least in my case) was NOT Swing for Scala **2.12**, it was Swing for Scala **2.11**.

If you try to use Swing for Scala 2.11 with Scala 2.12 you'll likely run into some weird errors. Ugh! Therefore, when you add dependencies to your *build.sbt* file, be careful to ensure that the dependency specs you copy from Maven are the correct (typically most recent) version.

By the way, if you ever need to see what version of Scala you're using, one way to check is simply by starting an interactive Scala shell, as used for REPL-style[52] interactive Scala processing:

Figure 7. Checking the Scala Version with the Interactive Scala Shell

```
$ scala
Welcome to Scala 2.12.1 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_121).
Type in expressions for evaluation. Or try :help.

scala> :quit
```

A nice overview of *sbt* written by Alvin Alexander can be seen **online**.[53]

For detailed online documentation about sbt, see the ***sbt Reference Manual***[54] in PDF format.

You may also be interested in traditionally published books about sbt, such as:

---

[50] https://docs.oracle.com/javase/tutorial/essential/environment/paths.html
[51] https://mvnrepository.com/artifact/org.scala-lang.modules/scala-swing_2.12/2.0.0
[52] https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop
[53] http://alvinalexander.com/scala/sbt-how-to-manage-project-dependencies-in-scala
[54] http://www.scala-sbt.org/0.13/docs/sbt-reference.pdf

1. Saxena's **"Getting Started with SBT for Scala"**[55]

2. Suereth and Farwell's **"sbt in Action: The simple Scala build tool"**[56]

That's enough talking about scala, swing, crypto, sbt and everything else -- let's get going and actually **build some code!**

[55] https://www.amazon.com/Getting-Started-Scala-Shiti-Saxena/dp/1783282673
[56] https://www.amazon.com/sbt-Action-simple-Scala-build/dp/1617291277

SECTION 2. Actually Building/Installing Code On The Local Machine

XI. Getting Started With Scala On The Mac: Prerequisites

As in earlier articles, we'll assume you're working on a current generation Mac (perhaps running MacOS Sierra). We also assume that your Mac is fully patched up-to-date and recently backed up.

- We'll also assume that you've already installed Apple's developer's tools, **Xcode.**[57]

- For the installation of many prepackaged libraries, we'll rely on **Brew.**[58]

- We also need Java. There are a confusingly large number of different versions, so to keep this simple, let me simply say that anyone who wants to try this little application should install **Java SE 1.8 version 131 JDK (also known as Java SE 8u131 JDK)**[59] or later. Please do this even if you've already got some other version of Java installed.[60]

XII. Installing Scala and sbt

Once you've gotten all the above prerequisites accomplished, it's then time to install Scala and sbt. Scala and sbt are easy to install on the Mac using brew, see Figure 8.

Figure 8. Installing scala and sbt using Brew

```
$ brew update
$ brew install scala
$ brew install sbt
```

Scala is a highly extensible language. While it may seem like we've already downloaded a "lot" of software, Scala and sbt will typically download still more (as our choice of features may require). Most often, additional required libraries will be retrieved from the **Maven repository.**[61] We'll talk more about sbt and Maven later in this white paper.

---

[57] https://developer.apple.com/xcode/downloads/
[58] https://brew.sh/
[59] http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html#javasejdk
[60] Note that **earlier** versions of Java (such as Java 1.7) will almost certainly NOT work. On the other hand, if you're reading this post months or years from when it was originally written and published, you should try to run the **latest** versions of Java that may have been released since that time.
[61] https://mvnrepository.com/tags/scala

XIII. Creating The Directory Structure We Need

sbt ("Simple Build Tool") has a specific directory structure it expects to see.

We'll create that directory structure using Unix shell commands modeled after those shown by Alvin Alexander's **"How to create an SBT project directory structure."**[62]

Let's call our project *SampleScala.* That implies having a project "home" or "base" directory by the same name:

Figure 9. Making Our Project's Top Level Directory

```
$ mkdir SampleScala
$ cd SampleScala
```

Now create the rest of the directory structure that sbt needs under that top-level "home" or "base" directory. *Be sure you're down in that directory before running the commands shown in Figure 10!*

Figure 10. Making the Rest of the Directories We Need

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

Having built the directory structure that *sbt* needs, we can visualize it with the *tree* command. Note that some users may first need to install *tree* with *brew:*

Figure 11. Installing tree

```
$ brew install tree
```

With *tree* installed, run the *tree* command from within the base *SampleScala* directory:

Figure 12. Running Tree

```
$ tree
```

[62] http://alvinalexander.com/scala/how-to-create-sbt-project-directory-structure-scala

tree should report a skeletal directory structure that looks like this:

Figure 13. Tree Output for Our Skeletal Scala Domain Structure

```
.
├── lib
├── project
├── src
|   ├── main
|   |   ├── java
|   |   ├── resources
|   |   └── scala
|   └── test
|       ├── java
|       ├── resources
|       └── scala
└── target
```

Note that the base *ScalaSample* directory (which is where we ran tree) is represented by the dot in the upper left corner of that diagram.

XIV. *SampleScala/build.sbt*

To get us closer to being read to use sbt to build our project, copy the **first** *build.sbt* file from Appendix I, putting it into the top level *SampleScala* project directory.

Note that the *build.sbt* file MUST be called *build.sbt* and MUST be placed in our base (top level) *SampleScala* directory.

If you look at the *build.sbt* file in an editor, be careful NOT to accidentally remove the blank lines included in that file. Those blank lines are present because sbt requires them in order for the file to be correctly interpreted and processed (e.g., no, that file did *not* "accidentally" somehow become "double-spaced")

XV. *SampleScala/project/assembly.sbt*

Copy *SampleScala/project/assembly.sbt* from Appendix III to *SampleScala/project/assembly.sbt*

We won't use that file immediately, but now's a reasonable time to get it created.

This file MUST be called *assembly.sbt* and MUST be put into *SampleScala/project/assembly.sbt*

XVI. *SampleScala/project/plugins.sbt*

Copy *SampleScala/project/plugins.sbt* from Appendix IV to *SampleScala/project/plugins.sbt*

Again, this is a file that will come into play later, but let's get it moved now, while we're doing other "housekeeping."

This file MUST be called *plugins.sbt* and MUST be put into *SampleScala/project/plugins.sbt*

XVII. Our Actual Scala Source Code

The next thing we need is our actual Scala program. Our program's a little under 1000 lines, so we've put that code into Appendix II.

That file MUST go into the *SampleScala/src/main/scala* directory and MUST use the name *SampleScala.scala*

XVIII. Creating Your DNSDB API Key File

Our sample program needs a Farsight API key in order to be able to access DNSDB. **Strict rules** pertain to the creation of that file:

- This file **MUST** be located in your *default directory* (e.g., on the author's Mac this is /Users/joe)

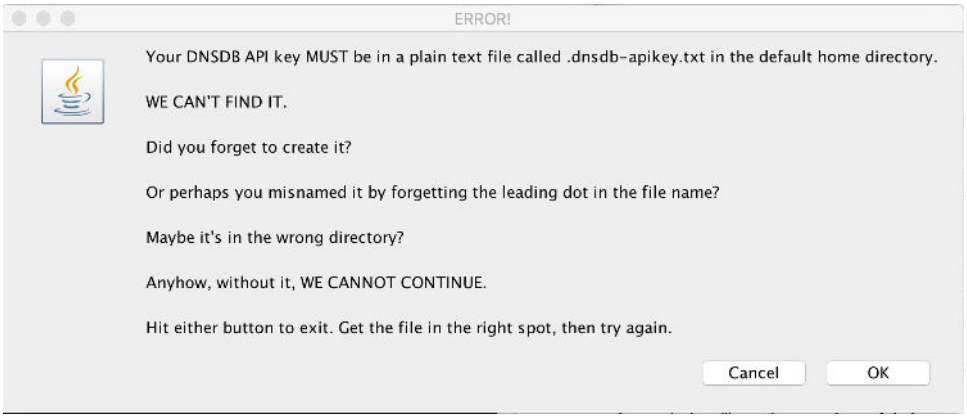
- It **MUST** be called *.dnsdb-apikey.txt* (NOTE the leading dot!)

- It **MUST** contain ONLY your 64 character DNSDB API key and nothing else (no quotes, no "comments," etc.)

- It **MUST** be a plain text file (not a MS Word document or other formatted file)

If you forget to enter your API key, or you didn't create that file in your default directory, the Scala program won't be able to run once we've built it. Instead, it will pop up a message like this one:

Figure 14. Missing API Key Dialog Box Message



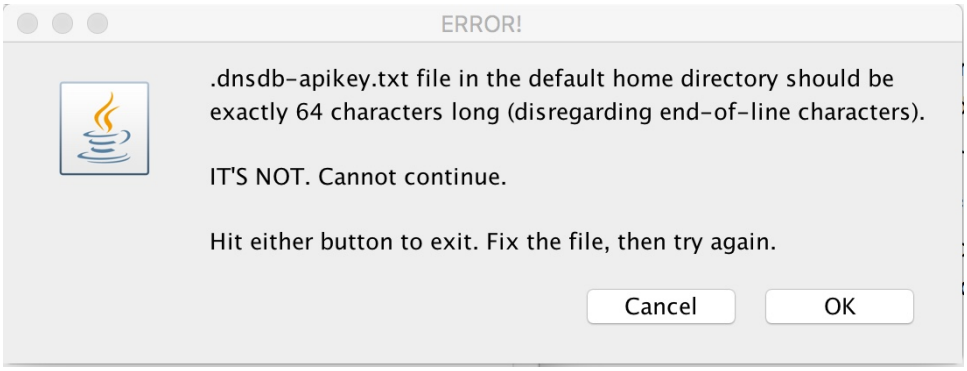
An easy way to create the .dnsdb-apikey.txt file on the Mac is with the nano editor:

Figure 15. Building A .dnsdb-apikey.txt file using nano

```
$ nano ~/.dnsdb-apikey.txt
[cut-and-paste or type in your API key here]
[control-o]
[control-x]
```

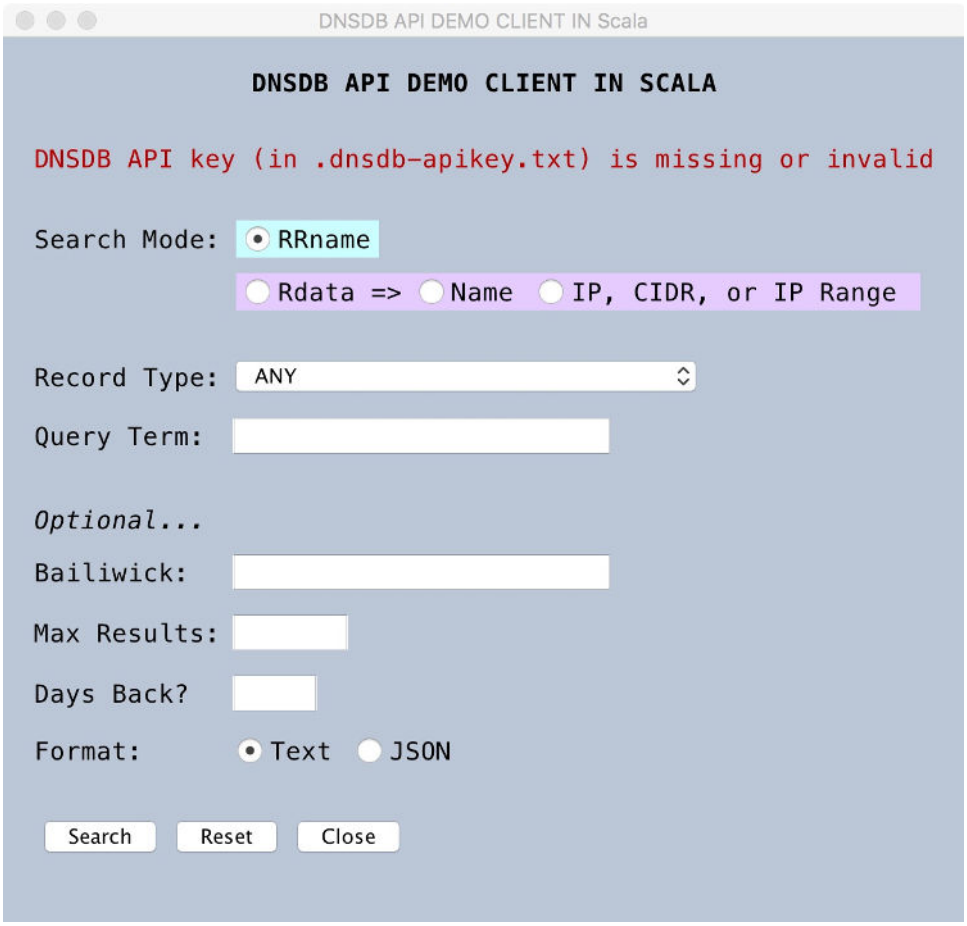
If you've accidentally omitted one or more characters, or you've accidentally added extra characters, we'll also catch that and refuse to run:

Figure 16. Wrong Length API key warning



If you manage to enter exactly 64 characters for your API, but still manage to make some sort of typo, when you try to run the program you'll see a different error (note the red text):

Figure 17. Bad API Key Warning



Or perhaps you simply don't ***HAVE*** a DNSDB API key? Please see the **Farsight Website**[63] for more information.

XIX: The JCE Unlimited Strength Jurisdiction Policy Files

In this step, you'll be downloading and installing the **Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files**[64](). The two jar files you need are bundled by Oracle into a single zip archive file.

Once you've download that archive and you've unziped the archive, copy the unzipped jar files into the **lib/security directory that's under your Java Runtime's home directory.**

[63] https://www.farsightsecurity.com/order-services/
[64] http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

Note: in our experience, many users, even tech-savvy users, often encounter problems successfully accomplishing this seemingly-simple task. If you're having difficult-to-resolve issues, consider building and running the check.java script from Appendix VI.

To do so, copy the Java commands listed in Appendix VI into the file check.java in your home directory. Then compile and run it by saying:

Figure 18. Running the Java Sanity Checking Script

```
$ javac check.java
$ java check
```

You should see a report that looks something like the following:

Figure 19. Java Sanity Checking Script Output:

```
Sanity checking DNSDB API Environment...
----------------------------------------

What operating system?...
  Mac OS X 10.12.4
What JRE?...
  1.8.0_131
Installed where?...
  /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/jre
Unlimited crypto policy files enabled?...
  true
Is there a .dnsdb-apikey.txt file in your home directory?...
Home dir: /Users/joe
Found: /Users/joe/.dnsdb-apikey.txt
Contents:
  [API key omitted here]
  Size=64 characters (disregarding EOL chars)?...
   true
```

STRONG CRYPTO FAQ ITEMS

"(a) How Do I Find My Java Runtime's Home Directory?"

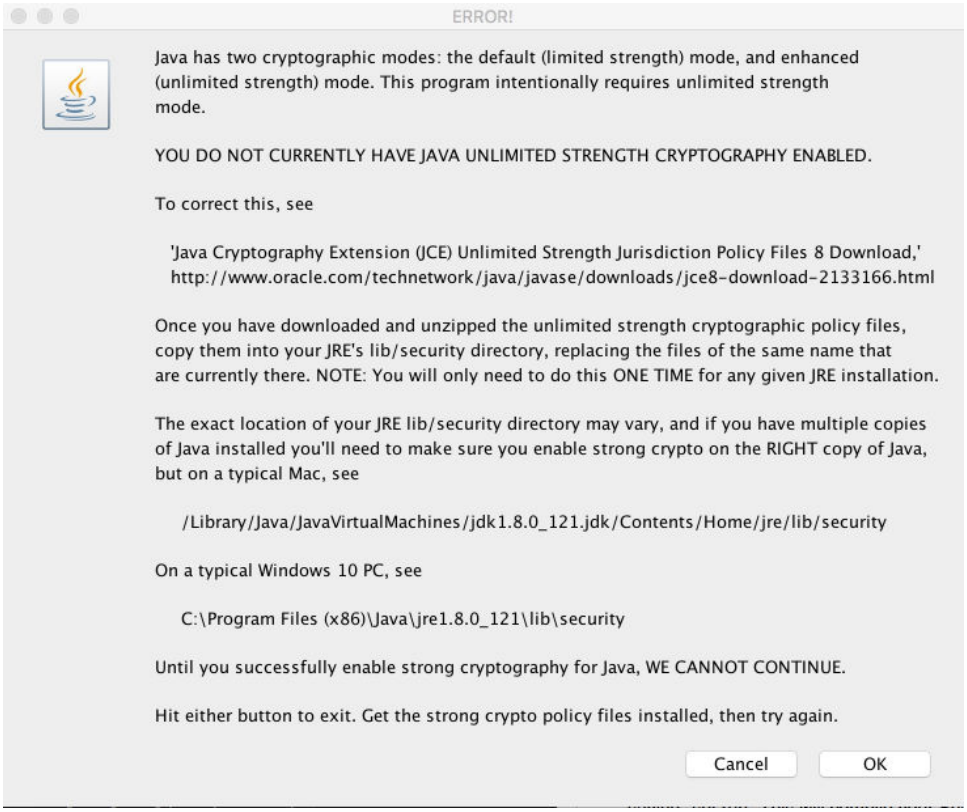
This can be potentially tricky, depending on whether or not you have multiple copies of Java installed. Many times people DO have multiple copies of Java installed, and as a result may end up installing the new strong crypto policy files in the wrong location. The Sanity Checking script mentioned in the preceding section may help.

Please note that it is **critical** for you to install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files in the **RIGHT** location (or, alternatively, in **ALL** possibly relevant Java locations).

If you DON'T install these policy files properly, our sample Scala GUI program won't work because we've **intentionally** specified a cryptographic suite that will **ONLY** work if you have the JCE Unlimited Strength Jurisdiction Policy Files installed.

If you try running the program WITHOUT the strong crypto policy files enabled, noting will happen except you'll see a "nudgy" dialog message that looks like:

Figure 20. Unlimited Crypto NOT Installed Error Message



To see possible locations where the Java crypto policy files may be installed, try searching with the Unix *find* command. To avoid reports of problems accessing some file, you may want to run the *find* command as super user using *sudo* (note that you'll need to supply an Administrator's password when prompted, if you're using *sudo*):

Figure 21. Searching Your System for local_policy.jar *file locations*

```
$ sudo find / -name local_policy.jar -print
```

In our case, we found (among other copies)...

Figure 22. Selected Sample local_policy.jar *Files Found Using the* find *Command*

```
/Users/joe/Downloads/UnlimitedJCEPolicyJDK8/local_policy.jar
```
(this is one of the unzipped files we downloaded from Oracle)

```
/Applications/Xcode.app/Contents/Applications/Application
Loader.app/Contents/itms/java/lib/security/local_policy.jar
```
(this is the copy of Java that comes with Xcode.app)

```
/Library/Internet
      Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/lib/security/local_policy.jar
```
(this is Java for use with Safari)

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/security/
local_policy.jar
```
(this is the installation we actually want to enhance)

So on our Mac, we then copied the *Downloads/UnlimitedJCEPolicyJDK8/*.jar* files over to */Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/security/*

Those various pathnames may be the right location for your Mac, too, or your Java runtime may be located somewhere else entirely. See the previously mentioned check script and the find command to turn up possible locations on YOUR system.

Note: You may need to use sudo or be root when copying or dragging-and-dropping those jar files, depending on the file and directory permissions of the destination.

(b) "What Happens If I DON'T/WON'T Install The Strong Crypto File?"

The sample GUI code won't work as coded.

(c) "Why Are You 'Forcing' Installation Of the Strong Crypto Files?"

This is a bit of "codliver oil," and it's for your own good. I'm a believer in encouraging deployment of strong cryptography, and this is one way for me to directly advance that goal in a Java-related context. Installing those files isn't THAT hard. You CAN do it, really.

(d) "What happens when I upgrade my existing version of Java to a new version sometime later?"

The unlimited strength crypto policy files will likely be over-written with default strength policy files, and the unlimited strength crypto policy files will need to be reinstalled.

XX. Building The Scala Code

One of the handy things about using sbt to build Scala programs is that you can compile and run our sample code by just changing down into the top-level project directory and then saying *sbt run* This will compile your Scala code, and if there are no errors, it will then proceed to automatically run it:

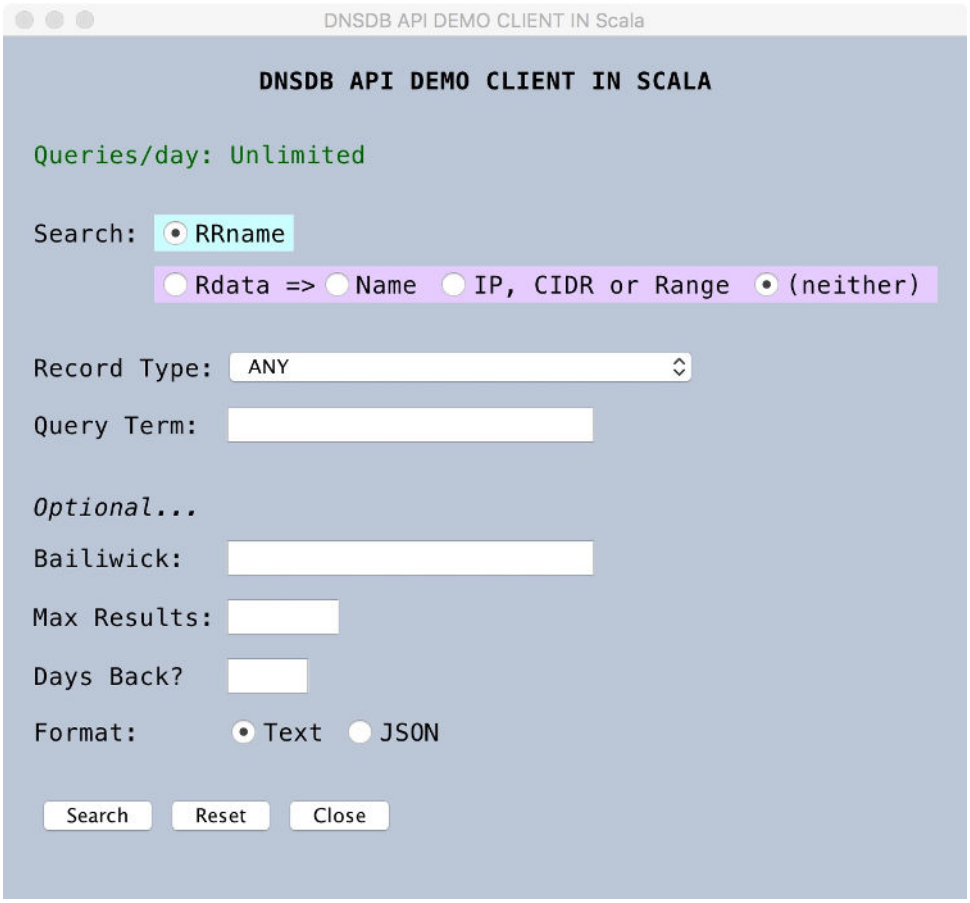
Figure 23. Compiling and Running the Scala Example

```
$ cd SampleScala
$ sbt run
```

The first time you do this, you'll probably see a bunch of output scroll by as sbt downloads additional libraries it requires. This process can take a minute or two, but will only need to be done once, so please be patient.

Once the Scala code has been compiled and begins to run, you should see a window that looks roughly like this:

Figure 24. What the GUI Interface Looks Like When Running

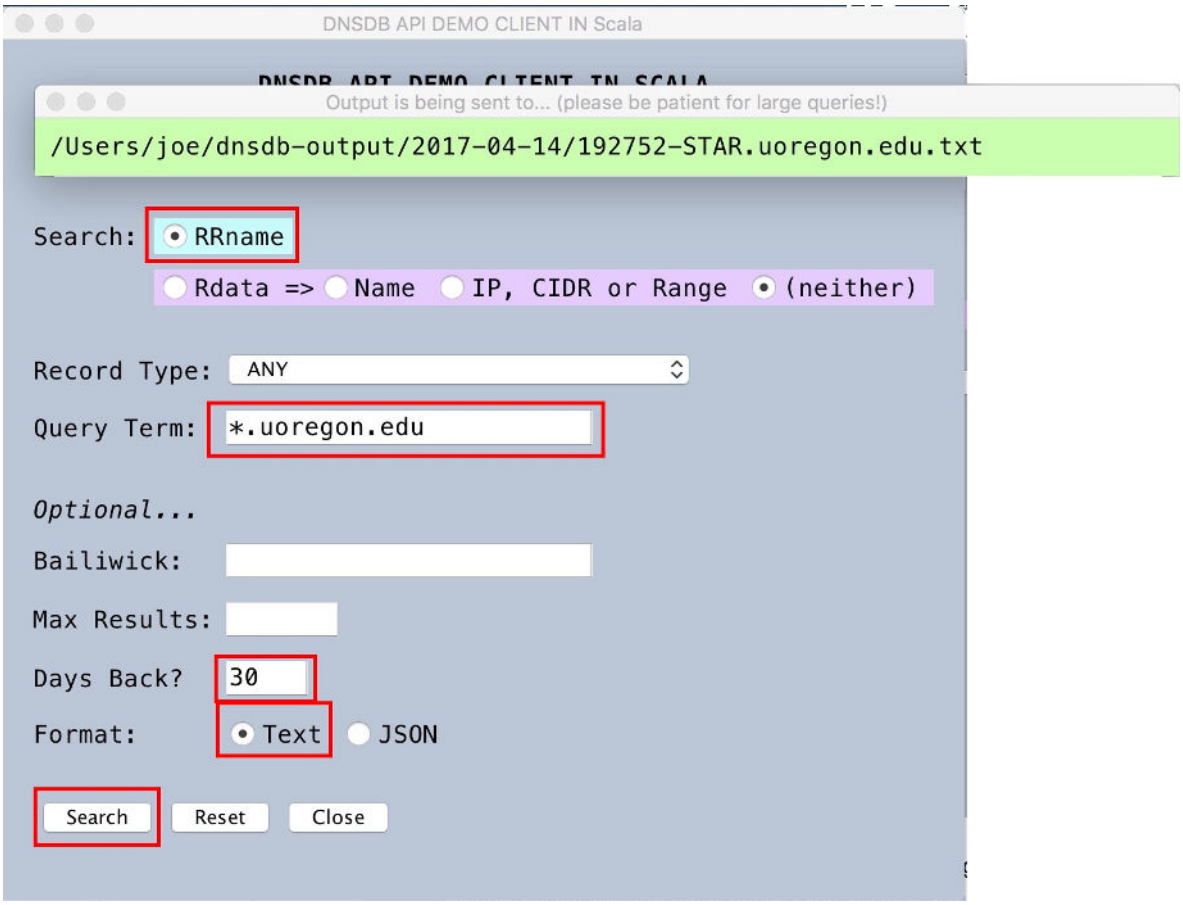


The GUI is now running! Good job! Note that the text in green near the top of the window will tell you how many queries you still have available at the time you start the session -- in this case, the client is running with an unlimited API key.

XXI. Running A Query and Looking At The Results

We can then try running a sample query. Let's find all RRname records known for **.uoregon.edu/ANY* for the last 30 days:

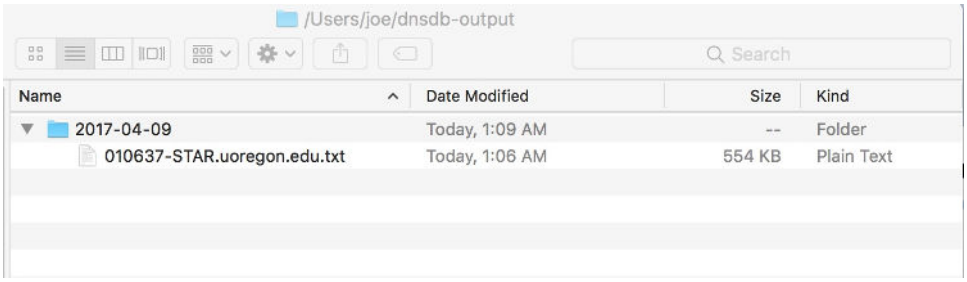
Figure 25. GUI Form Used To Make A Query for *.uoregon.edu *for the last 30 days*



NOTE: A window actually full of results will NOT be automatically be displayed!

We need to go explicitly look at our results when we're ready to do so... Use Finder to look in the dnsdb-output directory under your home directory, and choose today's date (or if you've previously done runs on another day, that earlier date). You can then double click on particular output files of interest.

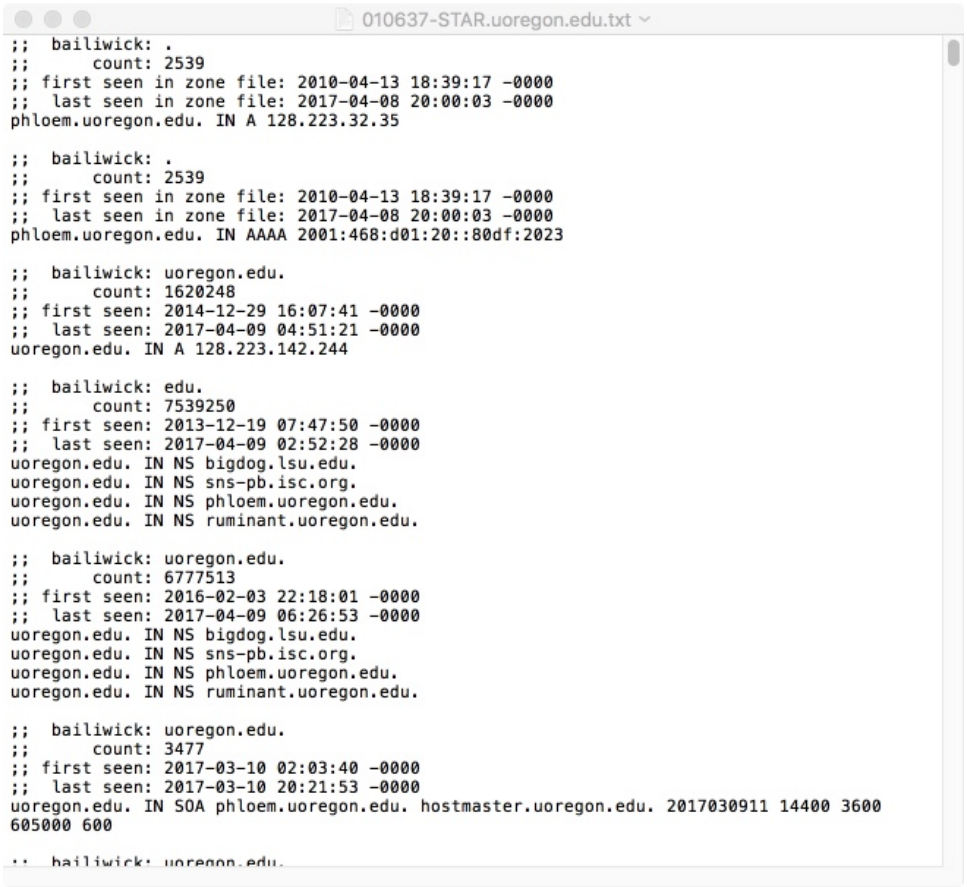
Figure 26. Choose An Output File (In This Case, We Only Have One Choice)



NOTE: To see the <u>CONTENTS</u> of these files, DOUBLE CLICK on it as you would any other text file.

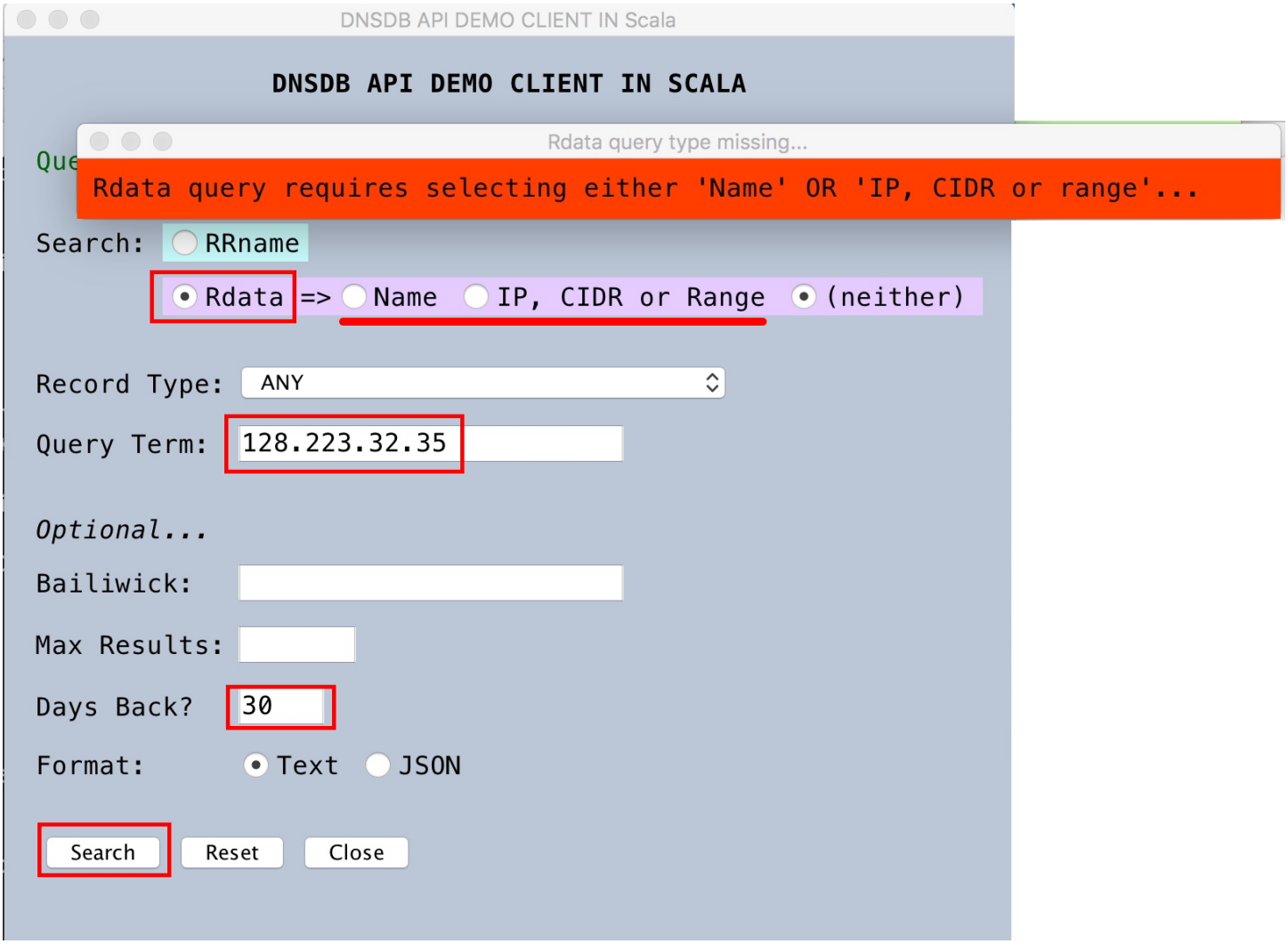
In this case, double clicking on that file shows us:

Figure 27. Text Format Results for One DNSDB Query



If you make an error, you'll be told about it. For example:

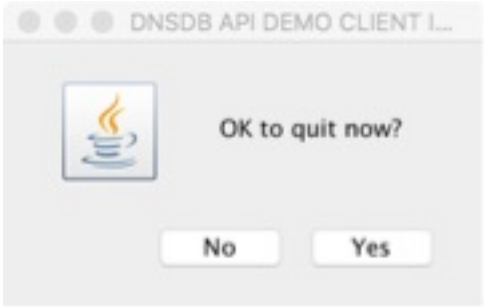
Figure 28. Sample Rdata error.



XXII. Quitting

To quit the sample Scala program, click the close button near the bottom of the Window. You should then see a small confirmation dialog box. Click "Yes" to quit.

Figure 29. OK To Quit Modal Dialog Window



XXIII. Packaging Our Code For Routine Use Without *sbt*

Normally you won't continue to run Scala programs within sbt once you're done developing them. Therefore, let's package up a jar file that we can run directly, instead. From the SampleScala base directory, say:

Figure 30. Creating A Portable jar *File with* sbt package

```
$ sbt package
```

Once that finishes running, copy *target/scala-2.12/samplescala_2.12-1.0.jar* to a convenient location, such as your home directory. You can then run it (assuming you have Java and Scala installed on your system) by saying:

*Figure 31. Running Our Potable Scala jar fil*e

```
$ scala ~/samplescala_2.12-1.0.jar
```

You can also obviously wrap that command into a trivial little shell script. For example, in my case:

Figure 32. Creating a Shell Script To Run The jar file

```
$ cd ~

$ nano query-gui
#!/bin/sh
/usr/local/bin/scala /Users/joe/SampleScala/target/scala-2.12/samplescala_2.12-
1.0.jar
[control-o]
[control-x]

$ chmod a+rx query-gui

$ sudo mv query-gui /usr/local/bin/.
```

Having set that script up, and assumng that /usr/local/bin is in your default path, you can then run the Scala graphical client just by sayng:

Figure 33. Running the shell script

```
$ query-gui
```

This is the end of what's strictly needed for you to use the GUI Scala DNSDB Client.

But what if you wanted to install and use this client on some OTHER Mac?

Or what if you wanted to use the graphical client on a PC running Microsoft Windows?

Let's go on to Section 3.

SECTION 3. Packaging For Use on 3rd-Party Macs and Windows PCs

XXIV. Creating A dmg Package For Use on A 3rd-Party Mac

To produce a **dmg**[65] file for use on other Macs, we'll use **sbt-native-packager.**[66]

To use *sbt-native-packager*, the *build.sbt* file in the project's base directory must include the line:

Figure 34. build.sbt enablePlugin command

```
enablePlugins(JavaAppPackaging)
```

If you copied your *build.sbt* file from the Appendix to this post, that line should already be present.

Likewise, if you followed the instructions in part two, you should already also have a file called `plugins.sbt` in `SampleScala/project` containing the line:

Figure 35. project/plugins.sbt

```
addSbtPlugin("com.typesafe.sbt" % "sbt-native-packager" % "1.2.0-M8")
```

Make sure you're in the SampleScala base directory, then run the packager using the two commands:

Figure 36. Running the packager to create the Mac disk image

```
$ sbt clean
$ sbt universal:packageOsxDmg
```

When the packager completes, it will have produced the file:

Figure 37. Name, Location and file details of the disk image file

```
SampleScala/target/universal/dmg/samplescala-1.0.dmg

$ ls -l samplescala-1.0.dmg
-rw-r--r--@ 1 joe  staff  7340032 Apr 26 13:28 samplescala-1.0.dmg

$ md5 samplescala-1.0.dmg
MD5 (samplescala-1.0.dmg) = 2831654e048319a8b5d9126376c53b2b

$ shasum -a 256 samplescala-1.0.dmg
9ebe47a3ed22732694b3248b44467444cd534dec4d733df02dcd1f90aef6b2b3
samplescala-1.0.dmg
```

Depending on whether anything has been updated since I built this disk image, your file size or checksums may vary from the ones shown here.

Anyhow, that's our package for the Mac.

---

[65] https://en.wikipedia.org/wiki/Apple_Disk_Image
[66] http://www.scala-sbt.org/sbt-native-packager/

XXV. Running Our dmg On A 3rd-Party Mac

To use our package on some other Mac, transfer the dmg file to the 3rd party system where you want to install the application. You can use a thumb drive, a file sharing site, or send it as an email attachment, etc.

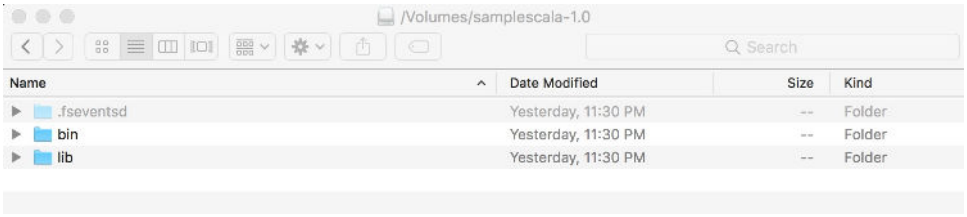
IMPORTANT NOTE: In order for that file to work on the third party system, the third party system user will still need to have Java 1.8 installed, AND they will need to have the Java strong crypto policy files installed, AND they will need to have a DNSDB API key installed, as described earlier in this post.

To run the interface on the third party system, open the *samplescala-1.0.dmg* package by double clicking on it in your Finder.

Select the resulting volume. (Not seeing volumes in your Finder window? Check to make sure *Finder --> View --> Show Sidebar* is enabled)

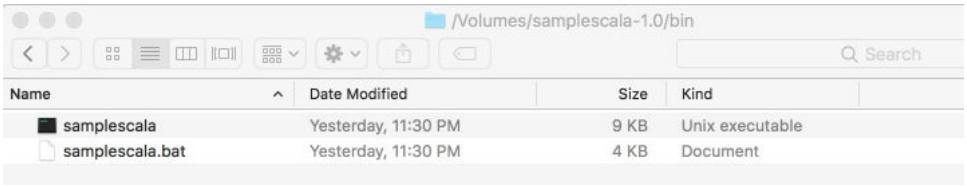
You should see a bin and lib subdirectory:

Figure 38. Inside our open'd sample-scala-1.0.dmg file



Go down into the bin subdirectory by double clicking on that folder.

Figure 39. Inside our bin directory



Double click on the sample scala Unix executable to run it. Proceed as "normal" from here on out.

XXVI. Making A Microsoft Windows Installer

While the preceding has all been done on the Mac, you can also run our little demo Scala GUI interface on Windows PCs. The Mac format *dmg* obviously won't work on a Windows PC, but we could build a PC *msi* installer that we could use under Windows.

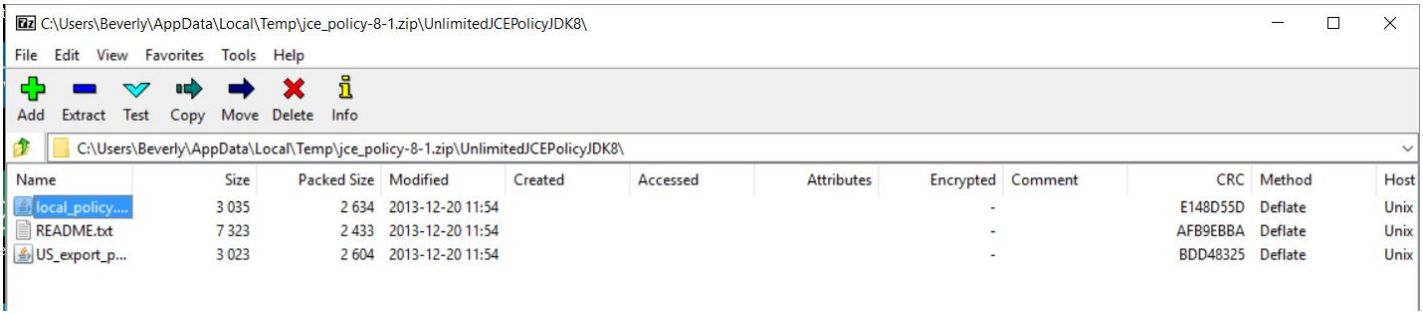
To produce a minimal *msi* installer for use under MS Windows, we'll use *sbt-native-packager*, just as we did to create our Mac *dmg* file. Note that in order to create the MSI file, **we need to create a Scala build environment on a Windows PC;** we cannot "cross package" our product for the Windows PC from our Mac.

Therefore, if you want to build an msi installer, dig out a convenient Windows PC, make sure it is patched up to date and fully backed up, and then install the Windows versions of

- **Java**[67]
- **Scala**[68]
- **sbt,**[69] and
- **The Java Unlimited Crypto Policy Files.**[70]

When it comes to installing the Java Strong Crypto Policy Files on a Windows PC, those files unzipped into the directory shown in Figure 40 (at least on our borrowed test system):

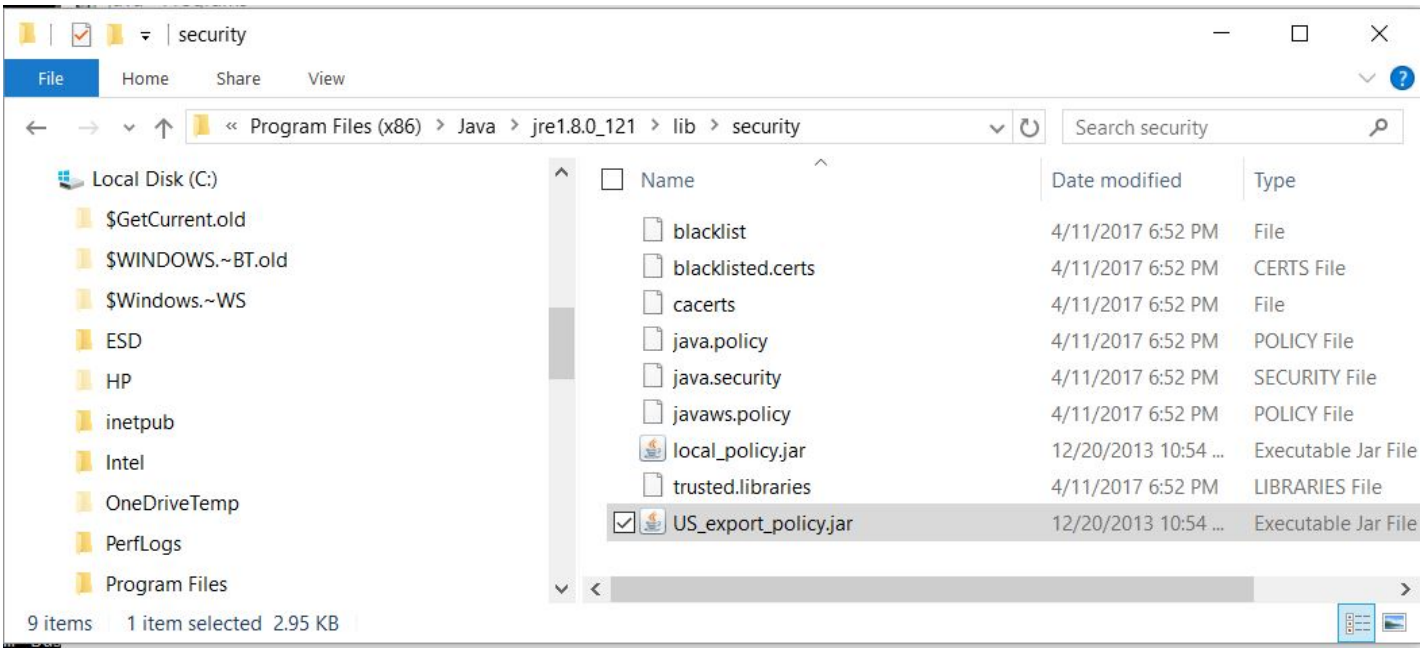
Figure 40. Unzipped strong crypto policy files (your location may vary)



[67] http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
[68] https://www.scala-lang.org/download/
[69] http://www.scala-sbt.org/download.html
[70] http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

We copied the jar files from that temporary directory over to
`c:\Program Files (x86)\Java\jre1.8.0_121\lib\security`

Figure 41. Destination for the strong crypto policy files (your location may vary)



Note: depending on the protective features active on your system, your system may be configured to limit your ability to copy those files to where you'd like to install them. If that occurs, use an account with administrator privileges, or see your local Microsoft system administrator for assistance.

We will also need to have your DNSDB API key installed in your default home directory in *.dnsdb-apikey.txt* just as we did on the Mac. Notepad or Word are fine for creating that file, just be **SURE** you save the .dnsdb-apikey.txt file as a **plain text** file!

Finally, you will also need one additional tool, and that's **Wixtoolset**.[71]

[71] http://wixtoolset.org/

For the purpose of building the Windows installer, ensure that you're using the 2nd *build.sbt* file provided in Appendix I!

Figure 42. Windows-Specific Bits of build.sbt

```
enablePlugins(JavaAppPackaging)

enablePlugins(WindowsPlugin)

// CHANGE THIS
maintainer := "Your Name <youremail@sample.com>"
packageSummary := "dnsdbAPIexample"
packageDescription := """Demonstrate a DNSDB API GUI Interface"""

// CHANGE THIS
// get new unique GUIDs via https://www.guidgen.com/
wixProductId := "GuidHere"
wixProductUpgradeId := "anotherGuidHere"
```

Note that you WILL need to edit that file to plugin YOUR name and YOUR email address on the maintainer line. (Those values are NOT authenticated and are NOT trustworthy, sadly, so don't be surprised if someday you run into Cyrano De Bergerac writing installers)

You WILL also need to generate unique GUIDs for your installer. An easy way to get those is by visiting **GUIDgen.com.**[72] GUIDs are a way of "fingerprinting" an installer so that Windows can determine whether or not it thinks it has already seen/installed a given package.

Now cd to the "project" directory (e.g., *SampleScala/project*). Be sure you've got a *plugins.sbt* containing the line:

Figure 43. SampleScala/project/plugins.sbt file

```
addSbtPlugin("com.typesafe.sbt" % "sbt-native-packager" % "1.2.0-M8")
```

This file MUST be in the specified directory and MUST have the specified name.

You may also want to create a license file in *SampleScala/src/windows/License.rtf* , at least if your first try at creating an *msi* installer generates a **"Lor ipsum"**[73] license rather than a blank window (or some license you actually like, whether that's an Apache 2.0 license, a BSD 2-clause license, or whatever).

You should now be ready to actually create the *msi* installer. From the *SampleScala* directory, run the packager with the commands:

Figure 44. Creating the `msi` installer

```
$ sbt clean
$ sbt windows:packageBin
```

[72] https://www.guidgen.com/
[73] https://en.wikipedia.org/wiki/Lorem_ipsum

When the packager completes after a minute or so, it will have produced the file *SampleScala\target\windows\SampleScala.msi*

Figure 45. Name, Location and file details of the disk image file

```
5,943,296 octets
MD5 = de36dd84e81b343473a61b5e03548bad
sha256 = f3acf1f499dc66b0e097c0d9a5041154d50557021a10ba30239f9c635a00ed08
```

That's the MS Windows installer for our little sample GUI application.

We've tested a copy of an *msi* installer built this way and it was been found to work on both Windows 10 and older Windows systems (e.g., things like Windows 7 systems).

XXVII. Using The MS Windows Installer On A 3rd-Party System

To use the *msi* installer on a 3rd-party system, transfer the msi file to the 3rd-party system where you want to install the application (by thumb drive, file share, email, etc.).

IMPORTANT NOTE #1: if you attempt to email that installer, some systems will routinely flag/block all executable content (including *msi* installers) as potentially malicious. Zipping the file before sharing may allow you to overcome some basic email content filters (obviously, this is not a very smart antimalware filter, if this "trick" actually works).

The *msi* file that we created was tested on **VirusTotal**[74] and scanned cleanly. You should always check any installer you make (or receive from someone else!) before trusting and executing it.

IMPORTANT NOTE #2: In order for the sample scala code to work on a third party system, the third party system user will still need to:

- Have Java 1.8 installed,
- Have the Java strong crypto policy files installed, and
- Have a DNSDB API key installed,

all as described earlier in this white paper.

[74] https://www.virustotal.com/

A typical run of the Windows installer will look something like the following...

Figure 46. Installer Opening Screen...

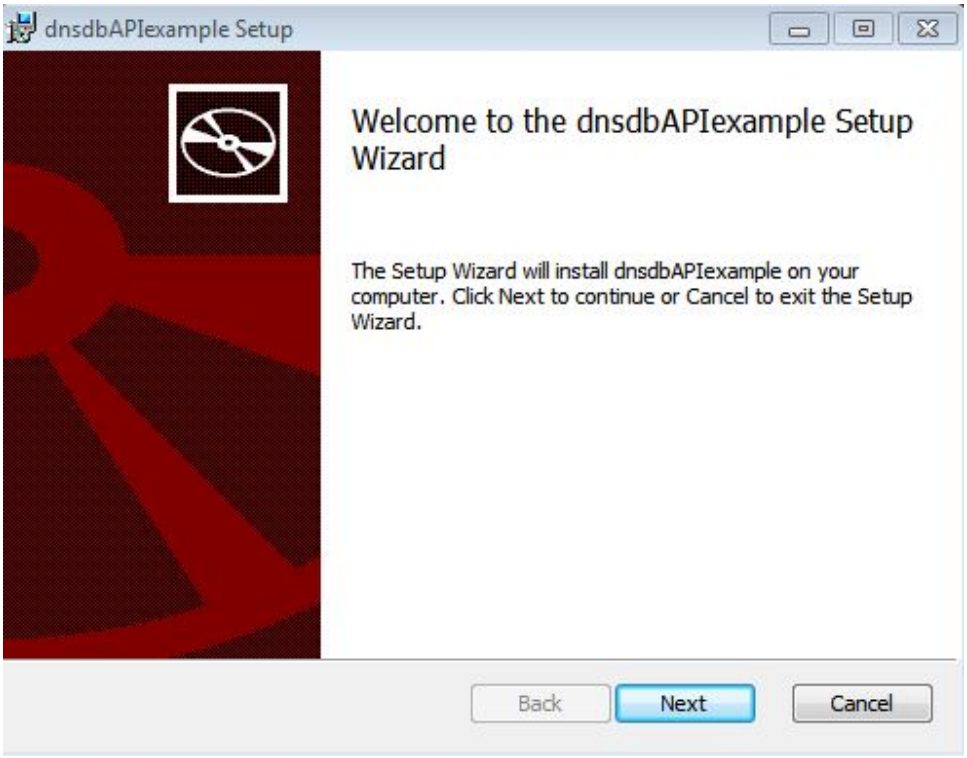


Figure 47. Blank License Screen (note that you must nonetheless click the "accept" box):

Figure 48. The custom setup box can just be clicked through (hit next):

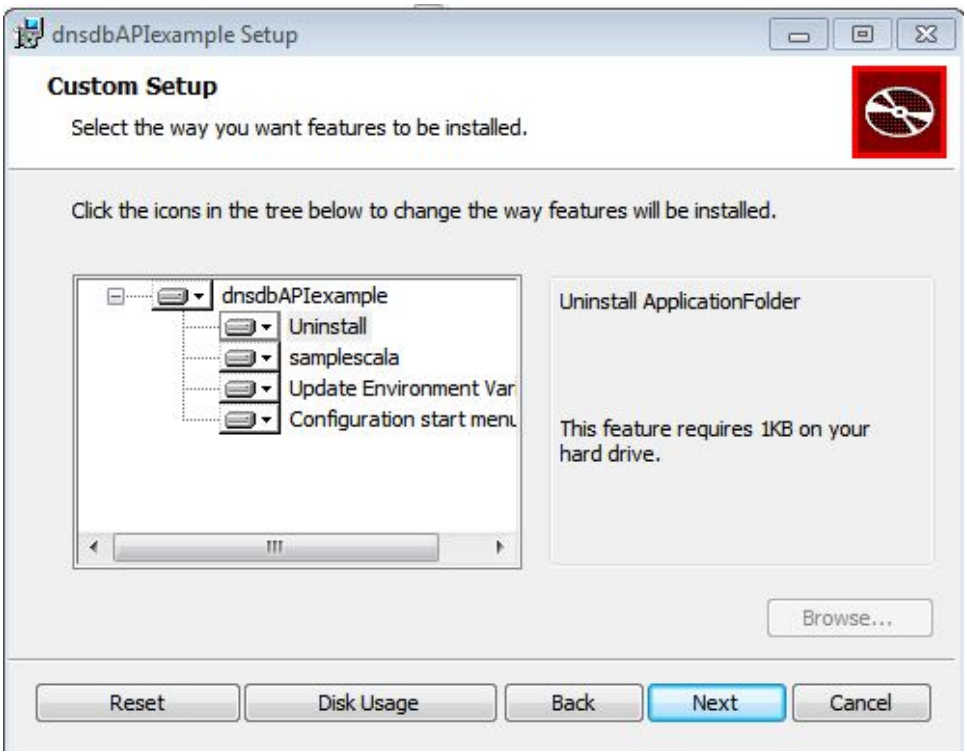


Figure 49. Now hit "Install" to do it...

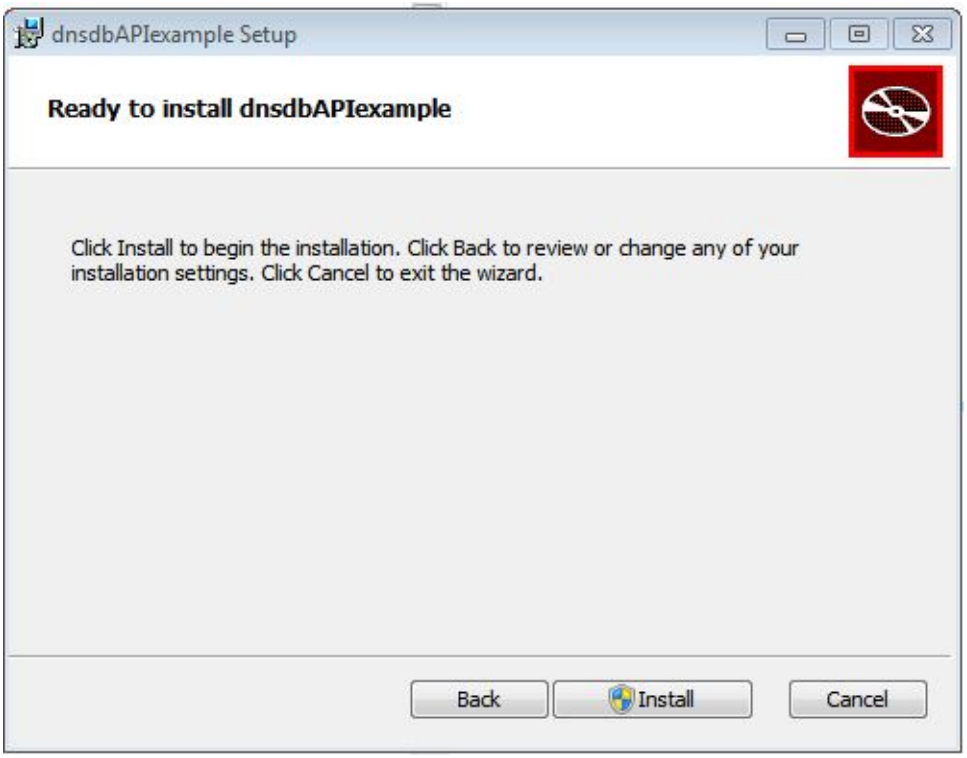


Figure 50. Because this isn't a signed executable (note the "unknown publisher") you'll typically see a warning:

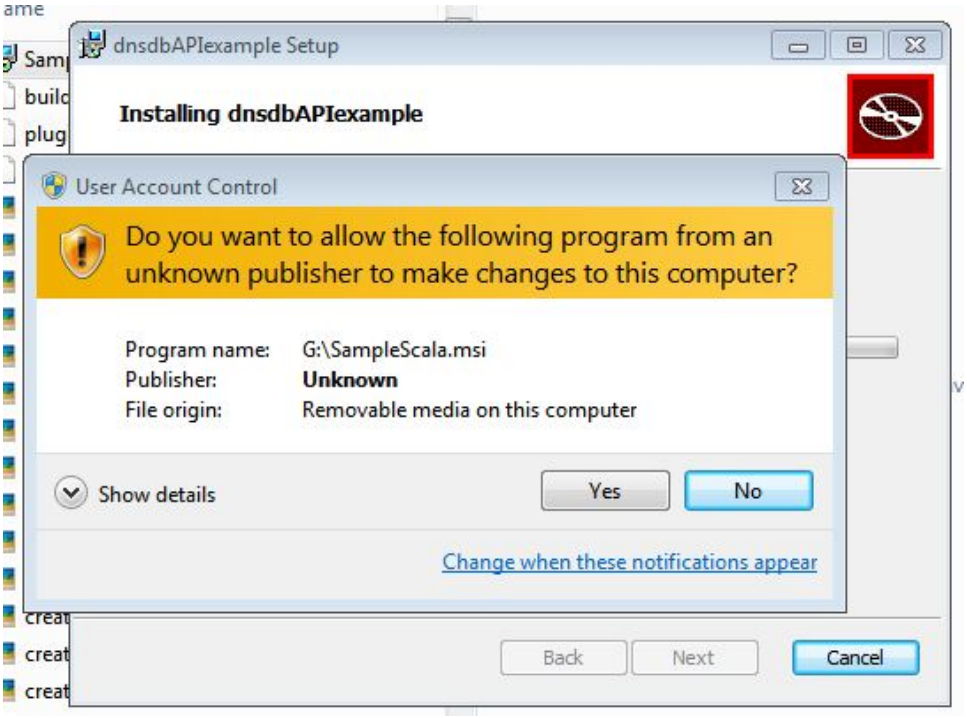


Figure 51. Once the installation completes, you will see:

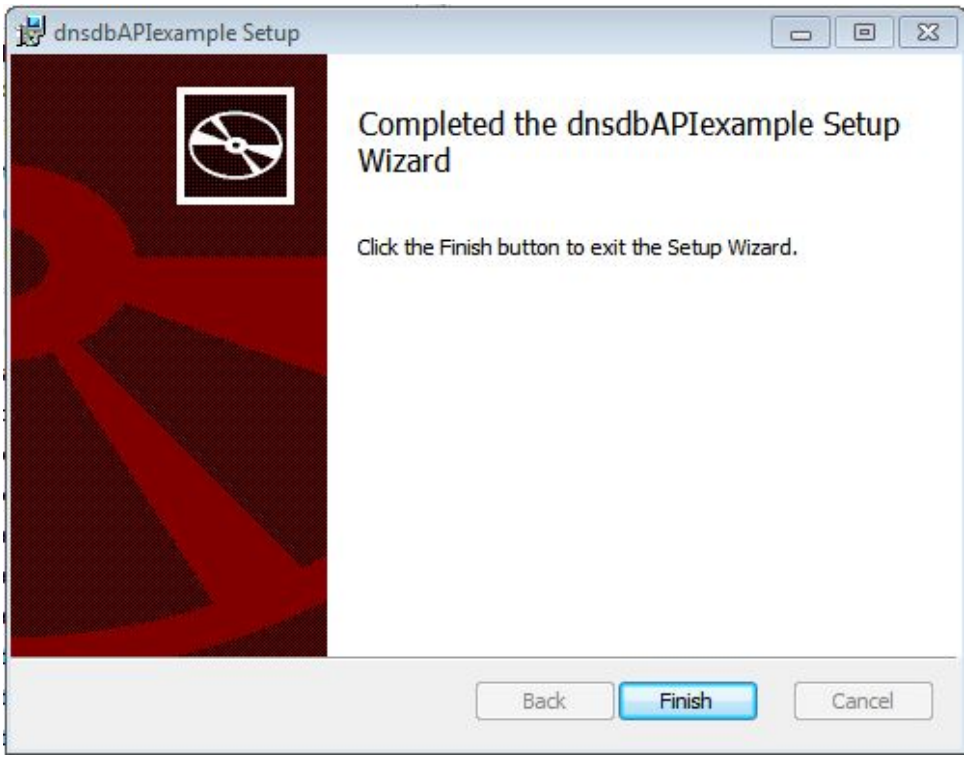


Figure 52. The executable is installed in the following location...

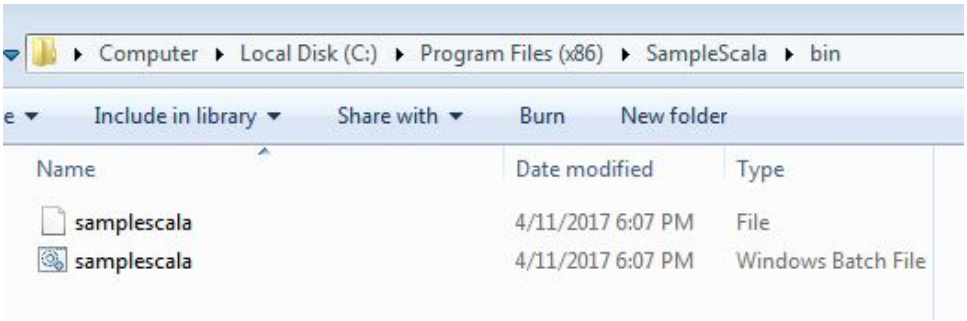


Figure 53. After clicking on the little "gear" icon associated with the batch file, our GUI interface will start:

Note: For reasons the author has yet to identify, there are subtle differences in how the RRname and Rdata hbox panels render color. This difference, while irritating, is purely cosmetic and doesn't negatively affect the usability of the program.

Figure 54. You can then try making a query just as we did on the Mac:

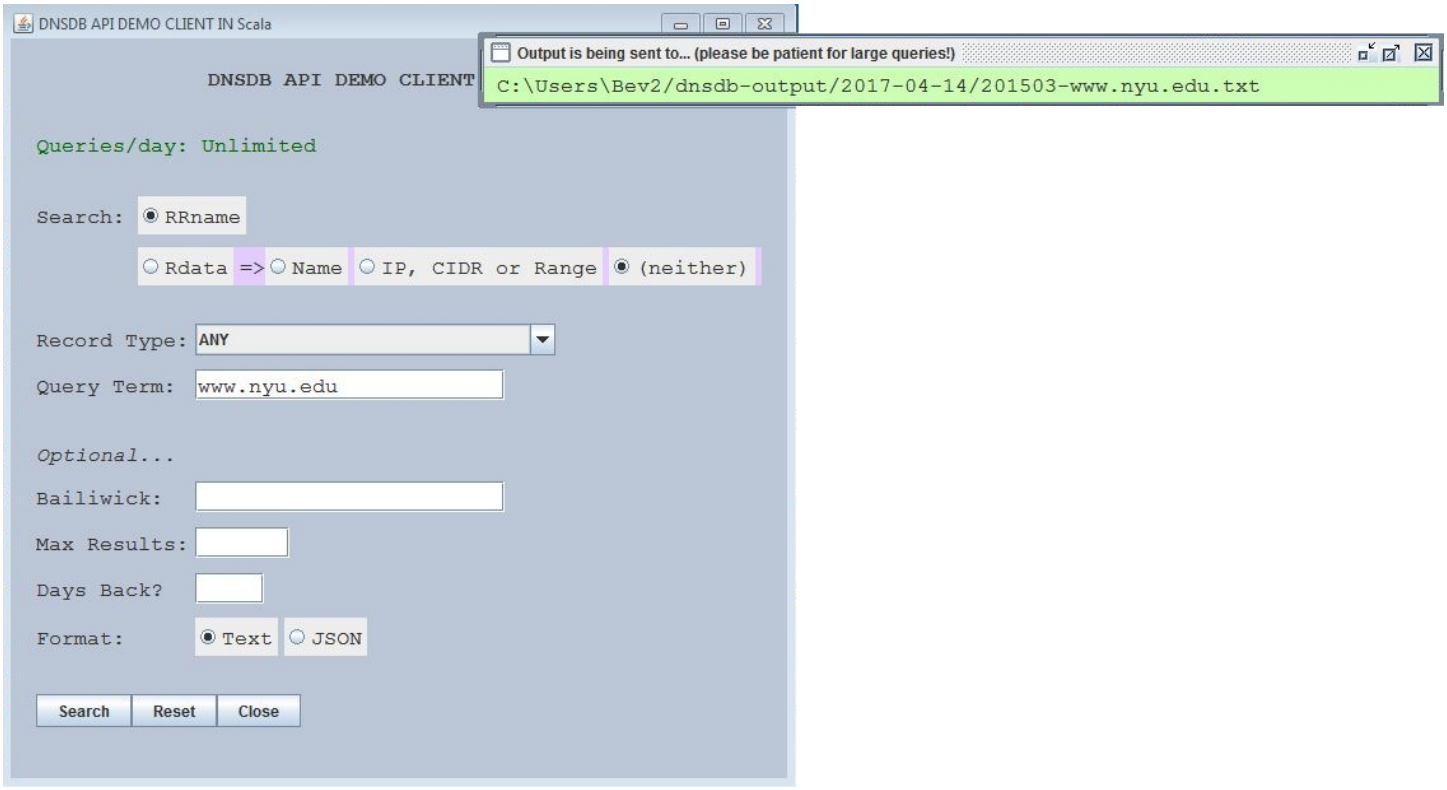


Figure 55. Output goes to a file in dnsdb-output, just under our default directory, with output saved under the current date as of the time the run was done:

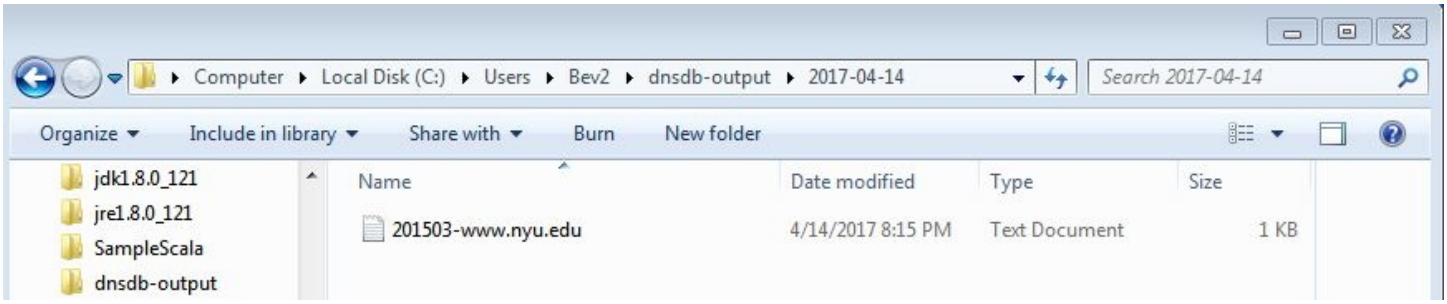


Figure 56. We can click on one of those files to see the contents:

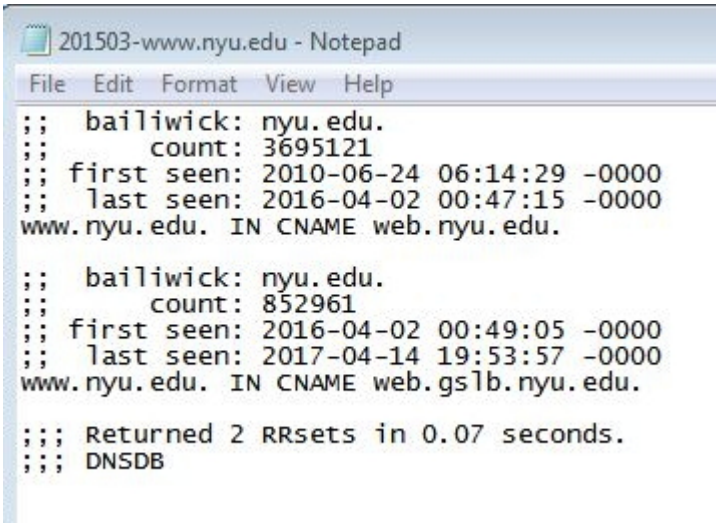
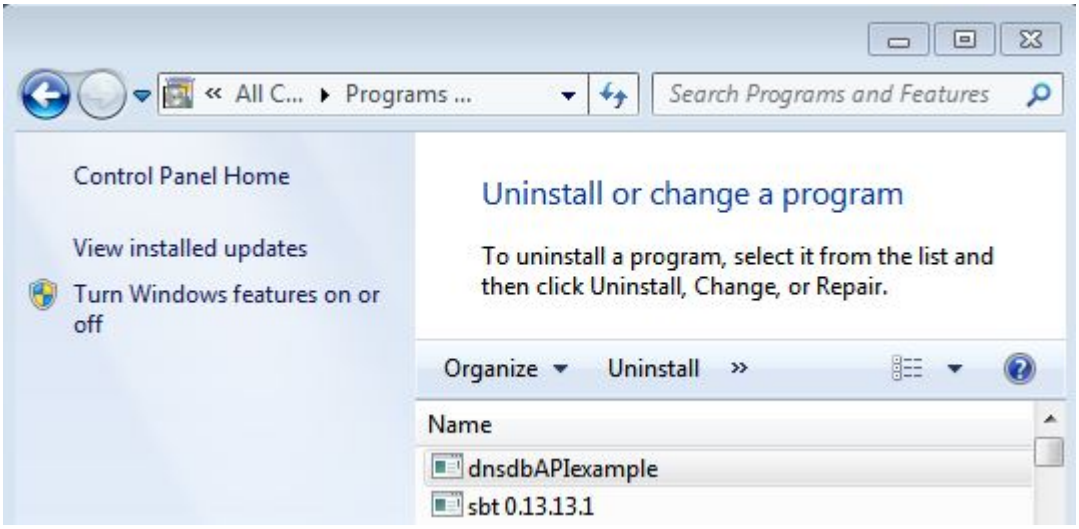


Figure 57. If you want to uninstall, there's even an uninstaller created by the `msi` installer (note the dnsdbAPIexample in the display)



XXVIII. Miscellaneous Notes

A few closing miscellaneous notes for those who may decide to try writing a bit of Scala themselves...

Licenses:

A number of the libraries or tools used in this article are subject to software license terms. If you elect to use the code shown here, please note that your use MUST compote with the licensing terms of those libraries or tools.

The *https://github.com/sbt/sbt-license-report* tool states that it can produce a report of applicable licenses.

Running this license reporting tool on my Scala project, I received the following license report.

Figure 58. License Report

samplescala-licenses

Category	License	Dependency	Notes
BSD	BSD 3-Clause	org.scala-lang # scala-library # 2.12.1	
BSD	BSD 3-clause	org.scala-lang.modules # scala-swing_2.12 # 2.0.0	

I've included a copy of the Scala license in Appendix V to this post.

Additional licenses to potentially be aware of (their applicability and impact may very depending on what you produce and how you use or distribute what you've built, but please note that I am not a lawyer, and that this is not legal advice):

- **Java SE Binary Code License**[75]
- **sbt License**[76]
- **Wixtoolset**[77]

[75] http://www.oracle.com/technetwork/java/javase/terms/license/index.html
[76] https://github.com/sbt/sbt/blob/1.0.x/LICENSE
[77] http://wixtoolset.org/about/license/

For its own open source programs and documentation materials, Farsight customarily applies an Apache 2.0 license. Thus, formally:

> *Copyright(c) 2017, Farsight Security, Inc.*
>
> *Licensed under the Apache License, Version 2.0 (the "License"); you may not use this code or document except in compliance with the License. You may obtain a copy of the License at*
>
> *http://www.apache.org/licenses/LICENSE-2.0*
>
> *Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.*

Cleaning Up the Formatting Of Your Scala Code:

If you're writing Scala code manually, it's easy to end up with gnarly-looking mis-formatted code.

scalafmt[78] makes it easy to turn a mess into readily-read-and-understood code (and it can be a lifesaver when it comes to tracking down missing or superfluous curly braces, too).

After installing the formatter as described at the scalafmt home page, save a copy of your original source file (just in case you accidetally manage to mess things up while using a new tool), then try:

Figure 59. Sample reformatting of Scala source code

```
$ scalafmt --stdin < src/main/scala/SampleScala.scala > src/main/scala/SampleScala.reformatted
```

Cleaning Up Unused Code (and Vars That Should Be Turned Into Vals, and Unneeded Imports, and Suboptimal Usages, and...)

If you experiment with various alternative approaches when building your Scala code, it's easy to end up with unused code, vars that could/should be rewritten as vals, and unneeded import statements among other things.

To find at least some of those, try running scalac (the Scala compiler) directly with the *-Ywarn-unused -Ywarn-unused-import -Xlint* flags set:

Figure 60. Sample quest for unneeded imports

```
$ scalac -Ywarn-unused -Ywarn-unused-import -Xlint src/main/scala/SampleScala.scala
```

[78] http://scalameta.org/scalafmt/

XXIX. What DIDN'T We Do?

There are a number of things that we could have done (but didn't do) in writing this code, including:

- The example didn't send output to a graphical window, just to a file. Sanely handling a graphical window that may contain up to a million observations is non-trivial.
- We used Swing instead of JavaFX.
- We didn't use a classical LayoutManager
- We made no attempt at interface internationalization (e.g., making a German or Russian or Chinese version of the app would require manual changes to embedded English string literals).
- We made no effort to support internationalized domains (e.g., no display of Cyrillic or Kanji domains using Punycode).
- We did not create integrated facilities for installing and managng API keys (we just assume you've create the required file in the specified location).
- We did not include an Edit menu with copy/cut/paste/select all/etc., nor GUI-accessible preferences for things like fonts, font sizes, foreground/background colors.
- We did not include GUI "rollover help" ("tool tips") nor online help/documentation at all for that matter.
- We only offer basic time fencing (just a simple "how many days back?" model)
- Nor did we include **raw hex rdata query**[79] support
- We did not build in support for proxies, for those who work in an environment where sending all traffic through a proxy server is required.
- We only did limited error checking/exception handling and reporting (we didn't bother catching and handling most potential errors)
- We didn't create any explicit unit tests
- We didn't create digitally-signed installers
- We didn't automatically resolve any missing dependencies as part of our installation process.

Maybe you'd like to try taking the starting code we provided, modifying it to address one or more of these deficiencies?

[79] https://www.farsightsecurity.com/2016/11/25/stsauver-dnsdb-rawhex/

XXX. Conclusion

Nonetheless, even with all those limitations, you've now seen an example of how to create a basic RESTful client in Scala for use with DNSDB.

There's obviously lots more that could be done to extend this example, but this white paper at least serves to illustrate the basics of building a DNSDB API GUI application, and the power of Scala. Maybe it will even tempt you to learn a new programming language, Scala?

If you have feedback, comments or questions about this article, feel free to contact the author at *stsauver@fsi.io*

Acknowledgements

The author gratefully acknowledges the reviews, suggestions, and testing assistance received from his Farsight colleagues, including (in alphabetcal order by last name) Mr. Ben April, Mr. Larry Pitman and Mr. Erik Wu.

Any/all residuals issues are the sole responsibility of the author.

Appendix I: The SampleScala/build.sbt file

If you only care about building the sample GUI application on the Mac, use this build.sbt file

```
name := "SampleScala"

version := "1.0"

scalaVersion := "2.12.1"

cancelable in Global := true

fork in run := true

connectInput in run := true

enablePlugins(JavaAppPackaging)

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

If you are on a PC running Microsoft Windows and want to build a Windows installer, use this build.sbt INSTEAD. Note that you will need to set some values as indicated below before you'll be able to use this file!

```
name := "SampleScala"

version := "1.0"

scalaVersion := "2.12.1"

cancelable in Global := true

fork in run := true

connectInput in run := true

enablePlugins(JavaAppPackaging)

enablePlugins(WindowsPlugin)

// CHANGE THIS
maintainer := "Your Name <youremail@sample.com>"
packageSummary := "dnsdbAPIexample"
packageDescription := """Demonstrate a DNSDB API GUI Interface"""

// CHANGE THIS
// get new unique GUIDs via https://www.guidgen.com/
wixProductId := "GuidHere"
wixProductUpgradeId := "anotherGuidHere"

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

Appendix II. SampleScala/src/main/scala/SampleScala.scala

```
package dnsdbexample

import java.io._
import java.net._
import java.text._
import java.util._
import javax.crypto.Cipher
import javax.swing._
import javax.swing.JFrame._
import scala._
import scala.io._
import scala.swing._
import swing._
import swing.event._
import Swing._

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

class UIoutput(var frameTitle: String, fileLocation: String, MyBoxColor: Color)
    extends 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 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
  }
}

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

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

  title = "DNSDB API DEMO CLIENT IN Scala"

  setDefaultLookAndFeelDecorated(true)

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

  val myFontSize = 16

  var myFontTitleSize = (myFontSize * 1.5)
```

```
// 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();

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

// DNSDB domain or IP
object myinput extends swing.TextField {
  columns = 24
  maximumSize = new Dimension(32, 24)
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}

// Bailiwick (used for RRname only)
object myinput2 extends swing.TextField {
  columns = 24
  maximumSize = new Dimension(32, 24)
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}

// Record data (used for Rdata only)
object myinput3 extends swing.TextField {
  columns = 24
  maximumSize = new Dimension(32, 24)
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}
```

```
// Max results to return
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 rrname = new RadioButton {
  text = "RRname"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}

val rdata = new RadioButton {
  text = "Rdata"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}

// only one or the other button can be checked
val mutex = new swing.ButtonGroup(rrname, rdata)

// .... 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 ....

// .... Radio button group

val nameinputmode = new RadioButton {
  text = "Name"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}

val ipOrNetworkMode = new RadioButton {
  text = "IP, CIDR or Range"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
}

val notSet = new RadioButton {
  text = "(neither)"
  font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
```

```
    selected = true
  }

  val mutex4 = new swing.ButtonGroup(nameinputmode, ipOrNetworkMode, notSet)

  // .... 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"))

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

  // When we start, make sure we have quota left...
  check_Remaining_quota

  // put up the main GUI form now
  do_GUI_form

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

  reactions += {
    case ButtonClicked(button) => {
      mutex.selected.get match {
        case `rrname` => {
          rrname.selected = true
          rdata.selected = false
          notSet.selected = true
          repaint()
        }
        case `rdata` => {
          rdata.selected = true
          notSet.selected = true
          repaint()
        }
      }
    }
  } // end of reactions

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

  def myClose() {
    val res = scala.swing.Dialog.showConfirmation(
      contents.head,
      "OK to quit now?",
      optionType = scala.swing.Dialog.Options.YesNo,
```

```
      title = title)
    if (res == scala.swing.Dialog.Result.Ok) sys.exit(0)
  } // end of myClose

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

  def APIfileDoesntExist() {
    contents = new BoxPanel(Orientation.Vertical) {
      background = c537
      // val res = scala.swing.Dialog.showConfirmation(
      scala.swing.Dialog.showConfirmation(
        contents.head,
        ".dnsdb-apikey.txt isn't in default home directory. Must Exit Now",
        optionType = scala.swing.Dialog.Options.YesNo,
        title = title)
      sys.exit(0)
    }
  } // end  of APIfileDoesntExist

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

  def reset_vals() {
    rectype.selection.item = "ANY"
    myinput.text = ""
    myinput2.text = ""
    myinput3.text = ""
    maxresults.text = ""
    gobackhowmanydays.text = ""
    mutex.select(`rrname`)
    mutex2.select(`textformat`)
    mutex4.select(`notSet`)
  }

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

  def check_Remaining_quota() {

    System.setProperty("https.cipherSuites",
                       "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384")
    System.setProperty("jdk.tls.ephemeralDHKeySize", "4096")

    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",
                            "Demo Scala DNSDB API Client v1.0 Quota Check")
    con0.setRequestProperty("Accept", "text/plain")
    con0.setRequestMethod("GET")
    con0.setConnectTimeout(3000)
    con0.setReadTimeout(3000)
    con0.connect()
```

```
    val responseCode0 = con0.getResponseCode()

    if (responseCode0 == 404) {
      // println("Quota check URL not found" + nl)
    } else if (responseCode0 == 400) {
      // println("Quota check URL formatted incorrectly" + nl)
    } else if (responseCode0 == 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 (responseCode0 == 500) {
      println("Error processing quota check request" + nl)
    } else if (responseCode0 == 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()
    }
  }

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

  def do_GUI_form() {

    System.setProperty("apple.laf.useScreenMenuBar", "true");
    System.setProperty("com.apple.mrj.application.apple.menu.about.name", "SampleScala");
    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

    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 = "DNSDB API DEMO CLIENT IN 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)
        contents += new swing.Label {

          if (limit0 == "unlimited") {
            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") {
            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 {
            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)
            }
          }

          font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)

        }
        contents += Swing.HGlue
        contents += Swing.HStrut(borderMargin)
      }
      contents += Swing.VStrut(bigVerticalSpaceBetweenItems)

      // Search Mode
      contents += new BoxPanel(Orientation.Horizontal) {
        background = c537
        contents += Swing.HStrut(borderMargin)
        contents += new swing.Label {
          text = "Search: "
          font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
        }
        contents += new BoxPanel(Orientation.Horizontal) {
          contents += rrname
          // this is a light blue color
          background = new Color(204, 255, 255)
        }
        contents += Swing.HGlue
        contents += Swing.HStrut(borderMargin)
      }
      rrname.selected = true
      contents += Swing.VStrut(10)
      contents += new BoxPanel(Orientation.Horizontal) {
        background = c537
        contents += Swing.HStrut(100)
        contents += new BoxPanel(Orientation.Horizontal) {
          contents += rdata
```

```
    contents += Swing.HStrut(5)
    contents += new swing.Label {
      text = "=>"
      font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
    }
    contents += nameinputmode
    contents += Swing.HStrut(5)
    contents += ipOrNetworkMode
    contents += Swing.HStrut(5)
    contents += notSet
    contents += Swing.HStrut(5)
    // this is a light purple color
    background = new Color(229, 204, 255)
  }
  contents += Swing.HStrut(borderMargin)
  contents += Swing.HGlue
  background = c537
}
contents += Swing.VStrut(bigVerticalSpaceBetweenItems)

// 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(verticalSpaceBetweenItems)

// Domain text box
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(borderMargin)
  contents += new swing.Label {
    text = "Query Term: "
    font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
  }
  contents += Swing.HStrut(5)
  contents += myinput
  contents += Swing.HGlue
  contents += Swing.HStrut(borderMargin)
}
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)

// Bailiwick text box
```

```
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(borderMargin)
  contents += new swing.Label {
    text = "Bailiwick:  "
    font = new Font(myFont, java.awt.Font.PLAIN, myFontSize)
  }
  contents += Swing.HStrut(5)
  contents += myinput2
  contents += Swing.HStrut(borderMargin)
  contents += Swing.HGlue
}
contents += Swing.VStrut(verticalSpaceBetweenItems)

// Max results
contents += new BoxPanel(Orientation.Horizontal) {
  background = c537
  contents += Swing.HStrut(borderMargin)
  contents += new swing.Label {
    text = "Max Results:"
    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(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") {
```

```
          restExample()
        }
        contents += swing.Button("Reset") {
          reset_vals()
        }
        contents += swing.Button("Close") {
          myClose()
        }
        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

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

      def setBoxLoc() {

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

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

      }

      // http://alvinalexander.com/blog/post/java/simple-https-example
      def restExample() {
        System.setProperty("https.cipherSuites",
                           "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384")
        System.setProperty("jdk.tls.ephemeralDHKeySize", "4096")

        // assume we're good to proceed; we'll then check...
        var goodToOutput = true

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

        // We need a query term. Did they specify one?
        if (myinput.text.length == 0) {
          val myNewWindow = new UIoutput("Query missing...",
                                         "Please supply a Query Term...",
                                         redTint)
          setBoxLoc()
          goodToOutput = false
          myNewWindow.location = new Point(xcoord, ycoord)
          myNewWindow.pack()
          myNewWindow.visible = true
        } else {
          // continue processing...

          // base RESTFUL URL
          if (rrname.selected == true) {
            httpsURL = httpsURL + "rrset/name/"
            // add on domain and recordtype
            httpsURL = httpsURL + myinput.text + "/" + rectype.selection.item
            // do we have a bailiwick? If so, add it on as well
```

```
  if (myinput2.text.length > 0) {
    httpsURL = httpsURL + "/" + myinput2.text
  }
} else if ((rdata.selected == true) &&
           (nameinputmode.selected == true)) {
  httpsURL = httpsURL + "rdata/name/"
  // add on domain and recordtype
  httpsURL = httpsURL + myinput.text + "/" + rectype.selection.item
} else if ((rdata.selected == true) &&
           (ipOrNetworkMode.selected == true)) {
  httpsURL = httpsURL + "rdata/ip/"
  var tempstring = myinput.text
  tempstring = tempstring.replace("/", ",")
  // add on IP or CIDR and recordtype
  httpsURL = httpsURL + tempstring + "/" + rectype.selection.item
} else if ((rdata.selected == true) &&
           (notSet.selected == true)) {
  val myNewWindow = new UIoutput(
    "Rdata query type missing...",
    "Rdata query requires selecting either 'Name' OR 'IP, CIDR or range'...",
    redTint)
  goodToOutput = false
  setBoxLoc()
  myNewWindow.location = new Point(xcoord, ycoord)
  myNewWindow.pack()
  myNewWindow.visible = true
}

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: 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
// target $HOME/dnsdb-output/$DATE/$TIME-$QUERY.txt

// 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 now = Calendar.getInstance().getTime()
val dateFormat = new SimpleDateFormat("yyyy-MM-dd")
val myDate = dateFormat.format(now)
homeDir = homeDir + "/" + myDate
val f: File = new File(homeDir)
f.mkdirs()

// now build the filename (time + querystring + rectype + .txt)
val timeFormat = new SimpleDateFormat("HHmmss")
val myTime = timeFormat.format(now)
var tempstring2 = myinput.text
// fix awkwardness in potential filenames
tempstring2 = tempstring2.replace("/", ",")
tempstring2 = tempstring2.replace(":", "#")
tempstring2 = tempstring2.replace("*", "STAR")
val outputPath = homeDir + "/" + myTime + "-" + tempstring2 + ".txt"
val f2: File = new File(outputPath)

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

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",
                       "Demo Scala DNSDB API Client v1.0")

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(3000)

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

        if (goodToOutput == true) {
          con.connect()
          responseCode = con.getResponseCode()
        }

        if (responseCode == 404) {
          val myNewWindow =
            new UIoutput("Query Error...", "No records found", redTint)
          setBoxLoc()
          myNewWindow.location = new Point(xcoord, ycoord)
          myNewWindow.pack()
          myNewWindow.visible = true

        } 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) && (goodToOutput == true)) {
          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) {
                // println(lines + nl)
                pw.write(lines + nl)
              }
            }

            // println(nl)
            pw.write(nl)

            in.close()
            pw.close()

          }
          // update available quota
          check_Remaining_quota
        } // end of UI
      }
    }
  }
}

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

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

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

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

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

class UIerror3 extends MainFrame {
  contents = new BoxPanel(Orientation.Vertical) {
    contents += new Label {
      text = " "
    }
  }
  val res = Dialog.showConfirmation(
    contents.head,
    "Java has two cryptographic modes: the default (limited strength) mode, and
enhanced\n(unlimited strength) mode. This program intentionally requires unlimited
strength\nmode.\n\nYOU DO NOT CURRENTLY HAVE JAVA UNLIMITED STRENGTH CRYPTOGRAPHY
ENABLED.\n\nTo correct this, see\n\n   \'Java Cryptography Extension (JCE) Unlimited
Strength Jurisdiction Policy Files 8 Download,\'\n
http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-
2133166.html\n\nOnce you have downloaded and unzipped the unlimited strength
cryptographic policy files,\ncopy them into your JRE\'s lib/security directory, replacing
the files of the same name that\nare currently there. NOTE: You will only need to do this
ONE TIME for any given JRE installation.\n\nThe exact location of your JRE lib/security
directory may vary, and if you have multiple copies\nof Java installed you'll need to
make sure you enable strong crypto on the RIGHT copy of Java,\nbut on a typical Mac,
see\n\n
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/security\n\nOn a
typical Windows 10 PC, see\n\n     C:\\Program Files
(x86)\\Java\\jre1.8.0_121\\lib\\security\n\nUntil you successfully enable strong
cryptography for Java, WE CANNOT CONTINUE.\n\nHit either button to exit. Get the strong
crypto policy files installed, then try again.",
    optionType = Dialog.Options.OkCancel,
    title = "ERROR!"
  )
  if (res == Dialog.Result.Ok)
    sys.exit(0)
  else
    sys.exit(0)
} // end of UIerror2

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

object SampleScala {

  var apiKeyFileStatus = "APIFileIsOK"

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

    // CHECK SANITY OF CONFIGURATION, FIRST...

    // 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 {
      // println("API Key File doesn't exist at .dnsdb-apikey.txt in default home
directory")
      val uiError2 = new UIerror2
      uiError2.visible = true
    }

    val fileContentsLength = myLine0.length()

    if (fileContentsLength == 64) {
      // file contents length is correct
    } else {
      // println("Wrong size : " + fileContentsLength)
      val uiError = new UIerror
      uiError.visible = true
    }

    // now check for the existence of strong crypto policy files
    val unlimited = Cipher.getMaxAllowedKeyLength("RC5")
    if (unlimited <= 256) {
      val uiError3 = new UIerror3
      uiError3.visible = true
    }

    // 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) {}
  }

}
```

Appendix III. SampleScala/project/assembly.sbt

```
addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.14.4")
```

Appendix IV. SampleScala/project/plugins.sbt

```
addSbtPlugin("com.typesafe.sbt" % "sbt-native-packager" % "1.2.0-M8")
```

Appendix V. Licenses

Scala

Scala is licensed under the BSD 3-Clause License.

Scala License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the EPFL nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS, "AS IS," AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Scala-Swing

Scala-Swing is also licensed under the BSD 3-Clause License.

Appendix VI. check.java source file

```
// portions from http://stackoverflow.com/questions/7953567/checking-if-unlimited-
cryptography-is-available

import javax.crypto.Cipher;
import java.io.*;
import javax.swing.filechooser.*;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileSystemView;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

public class check {
  public static void main(String args[]) throws Exception {
    System.out.println("Sanity checking DNSDB API Environment...");
    System.out.println("---------------------------------------");
    System.out.println(" ");
    System.out.println("What operating system?...");
    String osName = System.getProperty("os.name");
    String osVersion = System.getProperty("os.version");
    System.out.println("  " + osName + " " + osVersion);
    System.out.println("What JRE?...");
    String javaVersion = System.getProperty("java.version");
    System.out.println("  "+javaVersion);
    System.out.println("Installed where?...");
    String javaHome = System.getProperty("java.home");
    System.out.println("  "+javaHome);
    System.out.println("Unlimited crypto policy files enabled?...");
    boolean unlimited =
      Cipher.getMaxAllowedKeyLength("RC5") >= 256;
    System.out.println("  "+unlimited);
    System.out.println("Is there a .dnsdb-apikey.txt file in your home directory?...");
    JFileChooser chooser = new JFileChooser();
    FileSystemView view = chooser.getFileSystemView();
    String homeDir = view.getHomeDirectory().toString();
    System.out.println("  Home dir: " + homeDir);
    String checkFile = homeDir + "/.dnsdb-apikey.txt";
    File f = new File(checkFile);
    if (f.exists()){
      System.out.println("  Found: " + checkFile);
        System.out.println("  Contents:");
        String apiKeyFile = new String(Files.readAllBytes(Paths.get(checkFile)));
      apiKeyFile = apiKeyFile.replaceAll("(\\r|\\n)", "");
        System.out.println("  "+apiKeyFile);
        int fileContentsLength = apiKeyFile.length();
        System.out.println("  Size=64 characters (disregarding EOL chars)?...");
        if (fileContentsLength == 64) {
           System.out.println("  true");
      }
      else {
         System.out.println("  FALSE. Length ="+fileContentsLength);
        }
    }
    else {
      System.out.println(checkFile + "  NOT FOUND");
    }
  }
}
```