Tabulator Build and Hashing Instructions

Relevant to: NYRT120-29, NYRT120-35

URCVT v. 1.2.0 14-NY Tabulator Build and Hashing Instructions v. 1.0.0 document is solely for use in the State of New York. This document can be expanded or updated as is necessary or required. Any recommendations listed in this document should not supersede user jurisdiction procedures or other controlling governance entities.

URCVT v. 1.2.0 14-NY Tabulator Build and Hashing Instructions v. 1.0.0

Full build

To generate a build from the source code and then generate a hash of that build, first clone the rcv repository from GitHub.

Important note for Windows users: to ensure that the files in your copy of the repository exactly match the ones that were originally used to generate the Windows release build, you must have Git's autocrlf setting enabled before you clone the repository. This setting will cause Git to automatically convert the line endings in each source file from Unix-style (\n) to DOS-style (\r\n) when it clones the repository to your machine.

The instructions for cloning the repository are, therefore:

1. If Git is not already present on the machine, install it. The Windows version of Git is available here: https://gitforwindows.org/.
2. (If on Windows) Enable the autocrlf setting by running this command in the command prompt: git config --global core.autocrlf true
3. Clone the repository with this command: git clone https://github.com/BrightSpots/rcv.git

Building the application will also require the proper version of the Java development kit to be installed on the machine. Download and install the version of OpenJDK 14.0.1 (build 14.0.1+7) that corresponds to your machine's operating system (Windows) from this page: https://jdk.java.net/archive/ (Note that you must use this exact version of the JDK. If you obtain the JDK from a different location, even if the version and build numbers are the same, it may have subtle differences that will prevent your build from matching ours via the process we've developed. Specifically, we've determined that the Oracle-licensed build of OpenJDK is not identical to the London Jamocha Community CIC-licensed build.)

After installation (including setting your JAVA_HOME variable as detailed here (https://www.thewindowsclub.com/set-java_home-in-windows-10)), confirm that Java 14.0.1 is properly installed by running this command and verifying the version shown in the output:

```
java --version
```

Next, navigate into the top-level directory of the repository (rcv) and run a command to build the application using Java 14.0.1 and Gradle 6.5.1 (which is included in the rcv repository). The build will incorporate the RCTab source code along with all of the external libraries that it depends on (which are listed in the "Software Design and Specifications" documentation).

The Gradle build command on Windows:

```
gradlew.bat jlinkZip
```

The output of the Gradle build command is a single ZIP file called rcv.zip that will be located in the build subdirectory. To complete the build process, rename the ZIP file to reflect the platform and version of the application.

On Windows, building version 1.2.0:

```
rename build\rcv.zip build\universal_rcv_tabulator_v1.2.0_windows.zip
```

To generate a hash signature for the completed build, run this command on Windows:

```
certutil -hashfile build\universal_rcv_tabulator_v1.2.0_windows.zip SHA512
```

Example output on Windows:

```
SHA512 hash of build\universal_rcv_tabulator_v1.2.0_windows.zip:
c7704870b5a442ec9fb7212427377dfb822a8a9ac0d4310b68edfdd9e88fbd55821f962072a293e
deaf64529ef7288293dc9e854ed09d31627004c304f00f689
CertUtil: -hashfile command completed successfully.
```

Note, however, that two independent runs of the jlinkZip task will result in ZIP files with different hashes! Although the contents of the files in the two ZIP files will be identical, the timestamps of those files will not be; thus, the hashes will not match. As a result, generating a hash for the full ZIP file is only suitable for verifying that a trusted build has not been modified; it cannot be used to confirm that the file contents of one build match those of another.

To confirm that one build matches another regardless of timestamps, see "Comparing two builds" below.

Individual files

To generate hashes of all of the individual files in the source code, first, clone the repository according to the precise instructions outlined in the “Full build” section above.

Next, create a file called hash.bat one level up from the top-level rcv directory of the repository, paste the italicized script code below into hash.bat, save it, and then run hash.bat and check the results in the file all_hashes.txt:

```
@echo off

:: NOTE: This script must be placed one level up from the rcv directory

setlocal EnableExtensions EnableDelayedExpansion

set "INTEXTFILE=all_hashes.txt"
set "OUTTEXTFILE=all_hashes_temp.txt"

:: Delete the hash file if it exists already
if exist %INTEXTFILE% (
   del %INTEXTFILE%
)

:: Calculate the hash for every file here and in all subdirectories, appending to the file (format
"(filename) = (hash)")
for /r .\rcv %%f in (*) do (
   <NUL set /p ="%%f = " >> %INTEXTFILE%
   certutil -hashfile "%%f" SHA512 | findstr /v ":" >> %INTEXTFILE%
)

:: Replace the absolute paths to each file with relative paths (e.g. C:\temp\rcv => .\rcv)
set "SEARCHTEXT=%cd%"
set "REPLACETEXT=."
for /f "delims=" %%A in ('type "%INTEXTFILE%"') do (
   set "string=%%A"
   set "modified=!string:%SEARCHTEXT%=%REPLACETEXT%!"
   echo !modified!>>"%OUTTEXTFILE%"
)

del "%INTEXTFILE%"
rename "%OUTTEXTFILE%" "%INTEXTFILE%"

:: Sort the file
```

```
sort "%INTEXTFILE%" > "%OUTTEXTFILE%"
del "%INTEXTFILE%"
rename "%OUTTEXTFILE%" "%INTEXTFILE%"

endlocal
```

To see the expected output of running this script in Windows, refer to the "Source code file hashes" section below.

Comparing two builds

As noted above, two independent builds of the same source code will have different hashes due to the presence of file timestamps within the ZIP file. It is, however, possible to demonstrate that two builds are equivalent.

The process for generating a single timestamp-independent hash signature for a build ZIP requires a few steps: not only does the ZIP file store timestamps for the files it contains, but one of the files within the ZIP file, lib/modules, is, in turn, a collection of time stamped files. Fully decomposing the build to its individual constituent files thus requires using the jimage utility (part of the Java 14 development kit) to extract the files contained within lib/modules.

First, unzip the ZIP you built locally (by following the precise instructions in the "Full build" section above). Navigate to the top-level rcv directory and then run commands to extract lib/modules, generate hashes for all files except lib/modules, and generate a single hash based on that list of hashes. Be sure not to run the code in the unzipped rcv directory or execute any other commands that create, modify, or delete files within this directory, as any modifications to these files will change the result of the hashing process.

On Windows, create a file called hash.bat one level up from the rcv directory, paste the below in, save it, and then run hash.bat and check the results in the final command line output:

```
@echo off

:: NOTE: This script must be placed one level up from the rcv directory

echo Initiating batch hash procedure...

setlocal EnableExtensions EnableDelayedExpansion

set "HASHFILE=all_hashes.txt"
set "TEMPHASHFILE=all_hashes_temp.txt"
set "EXTRACTIONDIR=.\rcv\modules_extracted"
set "MODULESFILE=.\rcv\lib\modules"

if exist %HASHFILE% (
        echo Deleting existing hash file, %HASHFILE% ...
   del %HASHFILE%
)

if exist %EXTRACTIONDIR% (
        echo Deleting existing extracted modules directory, %EXTRACTIONDIR% ...
   rmdir /s /q %EXTRACTIONDIR%
)

echo Extracting contents of modules file...
mkdir %EXTRACTIONDIR%
cd %EXTRACTIONDIR%
jimage extract ..\..\%MODULESFILE%

echo Temporarily relocating modules file...
cd ..\..
copy %MODULESFILE% .
del %MODULESFILE%

:: Calculate the hash for every file here and in all subdirectories, appending to the file (format
"(filename) = (hash)")
echo Calculating hashes...
for /r .\rcv %%f in (*) do (
   <NUL set /p ="%%f = " >> %HASHFILE%
   certutil -hashfile "%%f" SHA512 | findstr /v ":" >> %HASHFILE%
)

echo Restoring modules file...
move .\modules %MODULESFILE%

:: Replace the absolute paths to each file with relative paths (e.g. C:\temp\rcv => .\rcv)

echo Replacing absolute paths with relative paths in hash file...
set "SEARCHTEXT=%cd%"
set "REPLACETEXT=."
for /f "delims=" %%A in ('type "%HASHFILE%"') do (
   set "string=%%A"
   set "modified=!string:%SEARCHTEXT%=%REPLACETEXT%!"
   echo !modified!>>"%TEMPHASHFILE%"
)
del "%HASHFILE%"
rename "%TEMPHASHFILE%" "%HASHFILE%"

echo Sorting the hash file...
sort "%HASHFILE%" > "%TEMPHASHFILE%"
del "%HASHFILE%"
rename "%TEMPHASHFILE%" "%HASHFILE%"

echo Calculating the hash of the entire sorted hash file...
certutil -hashfile %HASHFILE% SHA512

endlocal
```

Example output (Windows):

```
SHA512 hash of all_hashes.txt:
c832eb73c72eebe91dcb87ac4ea5daff089e3bcaaa88f756589f1a3e550a348080f430a661eed6f8
7e81bc54e362e54037f85e42b529bdc187becab62c409de7
CertUtil: -hashfile command completed successfully.
```

Document Revision History

Date	Version	Description	Author
04/24/2021	1.0.0	Tabulator Build and Hashtag Instructions	Louis Eisenberg