Using CAC certificates within Java applications

# Requirements

In order to be able to use certificates from a DoD CAC inside a Java application, you need to have:

* Java JDK/JRE that supports PKCS11 certificates (Oracle/Sun JRE/JDK supports Windows 32-bit, Linux 32/64-bit and Mac 64-bit)
* Apache ANT version 1.6 or higher
* A shared/linked library that can interface with a CAC reader (DoD libcackey works with Linux 64-bit and Windows 32-bit, I haven’t tested it with other systems)
* A CAC reader and valid card

Compiled versions and source code for “libcackey”, a CAC reader library, can be downloaded from Software Forge.mil here: <https://software.forge.mil/sf/go/projects.community_cac/frs.cackey> (NOTE: you will need a CAC and supported browser to access this site). The provided bundle also has the Windows 32-bit DLL and a compiled version of the Linux 64-bit .so file, as well as the latest source if you want to compile it for your system.

# Building the CAC access Java library jar file

1. Unzip the “cac-access-src.zip” bundle to a directory
2. From a console in the directory where you unzip the bundle, execute “ant” and it should build the jar file.
3. The jar file will be called “cac-access-1.0.jar” and located in the “dist” directory where you unzipped the bundle.

# Using the CAC access Java library

## Adding the library to your application’s classpath

Once you have successfully built the CAC access library, all you have to do is add it to your application’s Classpath to start using it, AND specify 1 System property. For example:

java -Dcac.library=<path to libcackey> -cp cac-access-1.0.jar:<the rest of your application’s classpath> <your application’s Main/entry>

The “cac.library” system property needs to be set to the absolute path location to the CAC reader shared/dynamic link library. On Windows, the file is “libcackey.dll”, on Linux, the file is “libcackey.so”. Just make sure the property is set to the FULL PATH to the library file, including the filename.

There is another system property you can set, “cac.slot”. This property value defaults to “0”. You only need to set this if your system does not initialize your CAC reader and ID certificate to be in the first slot. Try using the library without setting this property first. If you get something like a “PKCS11 not found” exception, you may have to set this property value to “1” or “2”.

## Enabling CAC access from within your application

To “turn on” the CAC access, you only need to import the access class and invoke a single static method from within your application. Add these lines:

import org.emerkle.cac.CACAccess;

CACAccess.init();

The Java code catches almost any Exception that might be thrown and logs them to the console, so you don’t have to wrap the call with a try-catch block. You may want to put logic in your application to determine if you should even try to make a call to the “init()” method.

## Entering your CAC PIN

If you get the library integrated into your application, calling the “init()” method will prompt you for your CAC PIN on the console. Your PIN will not be echoed to the console, and will only be asked for once for the life of the running application. This means you will not be able to use the library as-is without launching it from a console. It should be fairly easy to modify the source to add access routines that prompt for your CAC PIN by other means (i.e. a GUI input field, a system property, etc).

# Java Source Bundle

The provided Source bundle contains the following:

Length Date Time Name

--------- ---------- ----- ----

0 2011-10-20 13:23 build/

0 2011-10-20 13:24 build/classes/

0 2011-10-20 13:24 build/classes/mil/

0 2011-10-20 13:24 build/classes/mil/geocent/

0 2011-10-20 13:24 build/classes/org/emerkle/cac/

3093 2011-10-20 13:24 build/classes/org/emerkle/cac/CACAccess.class

2502 2011-10-20 13:24 build/classes/org/emerkle/cac/CACConfig.class

1465 2011-09-23 02:20 version.properties

371 2011-09-23 09:22 .classpath

0 2011-10-20 13:24 dist/

4256 2011-10-20 13:24 dist/cac-access-1.0.jar

368 2011-09-23 09:22 .project

0 2011-10-20 13:22 src/

0 2011-10-20 13:22 src/mil/

0 2011-10-20 13:22 src/mil/geocent/

0 2011-10-20 13:22 src/org/emerkle/cac/

2746 2011-10-06 16:22 src/org/emerkle/cac/CACAccess.java

2955 2011-10-06 16:22 src/org/emerkle/cac/CACConfig.java

0 2011-10-20 13:23 resources/

0 2011-10-20 13:26 resources/linux\_64bit/

94008 2011-10-20 13:26 resources/linux\_64bit/libcackey-0.6.5.so

0 2011-10-20 13:22 resources/windows\_32bit/

132856 2011-10-06 15:47 resources/windows\_32bit/libcackey-0.6.5.dll

52915 2011-10-20 13:23 resources/commons-logging-1.1.jar

680 2011-09-23 02:20 log4j.xml

2356 2011-10-20 13:24 build.xml

--------- -------

300571 26 files

The source code is only 2 files in “src/org/emerkle/cac”. You shouldn’t have to do anything to CACConfig.java. You may want to modify CACAccess.java if you want to be able to prompt for a CAC PIN from something other than the console.

The “.project” and “.classpath” files are for use with Eclipse. You may need to modify these a bit to work in Eclipse.

The “build.xml” file is the ANT build script. It should work as-is.

The “resources” directory contains a pre-compiled Linux 64-bit libcackey library, a Windows 32-bit precompiled libcackey library, the libcackey library source and the Apache commons logging library (for application logging). If neither of the pre-compiled libraries seems to work, you can try compiling from the provided source, or downloading a pre-compiled library form Software Forge.mil (<https://software.forge.mil/sf/go/projects.community_cac/frs.cackey>)