Connecting to a Bugzilla, JIRA server with a self-signed SSL certificate

This article applies to: *Deskzilla 1.x-2.0, JIRA Client 1.x-2.0, and connecting to Bugzilla and JIRA servers via https://... connections.
For clarity, this article is written about JIRA Client and JIRA, but it applies to Deskzilla / Bugzilla as well.

This **does not** apply to JIRA Client version 2.1 and later and Deskzilla version 2.1 and later, as the issue has been solved.

Problem

The problem is that JIRA Client cannot connect to JIRA over a secure connection (for example, to the url https://jira.company.com), with the following error text:

```
Connection problem: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to registered target
```

This reason for this problem is that the SSL certificate used on the server is self-signed and not trusted by default by the underlying Java security platform.

> You might not experience this problem, even if the certificate you use is self-signed. Do not apply this solution if JIRA Client has no problem connecting to JIRA.

> The self-signed certificate should have Common Name (CN) equal to the host name of the JIRA web site, otherwise the solution won't work! You can inspect Common Name among other certificate's properties when viewing it in a browser.

Solution

The solution involves getting the untrusted SSL certificate from the server and making it trusted by importing it into a special file, java keystore.

1. Locate Java in Use

If you downloaded the default JIRA Client distribution, it contains bundled JRE (Java Runtime) and when you run *jiraclient.exe*, it looks for the bundled JRE in the *installFolder*\jre. On Windows Vista 64-bit, the default location of the program is *C:\Program Files (x86)\JIRA Client*, and so the location of JRE will be *C:\Program Files (x86)\JIRA Client\jre*.

If you downloaded JIRA Client without bundled Java, or run it with .bat / .sh script, then you probably should know which Java does JIRA Client use, or you can find out by inspecting your PATH and JAVA_HOME variables.

2. Locate *cacerts* Key Store File

The location of *cacerts* is *jreHome/lib/security/cacerts*, where *jreHome* is the home path of the JRE in use. (Note that if you're using a JDK, then *jreHome* would be *jdkHome/jre*.)

3. Export Server Certificate

Now you need to get a file with exported server certificate. If you have Firefox browser installed, you can do the following:

1. Open the JIRA web site in Firefox (if needed - go through the warnings and add site's certificate as an exception).
2. Double-click on the "lock" icon in the status bar.
3. Click on "View Certificate" in the Web Site Identity section
4. Click on Details tab and then on Export button
5. Select a .crt file to export your key, use the default X.509 PEM format.

4. Import Certificate into the Key Store

Now you can import the server's certificate into the located *cacerts* file. Change into the directory where cacerts is located and run *keytool* command, located in "bin" subdirectory under *jreHome*:

```
..\..\bin\keytool -import -file path/to/the/exported/file.crt -alias my_jira_server -keystore cacerts
```

The default password is **changeit**, unless you have changed it.

Keytool will ask for confirmation, to which you should answer **yes** and then write updated store to the disk.

You have to have write permissions to the cacerts file and its directory. If needed, start command line with Administrative permissions under Windows, or sudo under Linux or Mac OS X.

5. Start JIRA Client

Try to connect. If it doesn't work, double-check that

- file cacerts got modified;
- cacerts contains your server's certificate (use *keytool -list -keystore cacerts* and look for alias my_jira_server);
- you're running JIRA Client with the JRE that has cacerts modified (not some other JRE on your hard drive).

If all looks correct but it doesn't work, please contact support.

Other solutions are also possible, for example, setting Java key store to a different file with certificate already there.