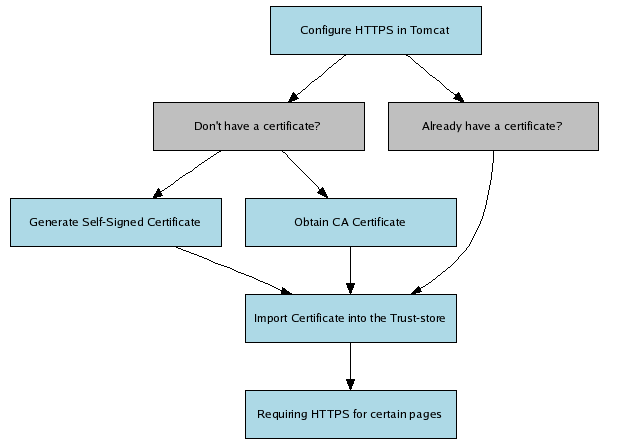
JIRA SSL CONFIGURATION:

<https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-ImportCertificateIntoTheTruststore>

# [Running JIRA over SSL or HTTPS](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS)

## **Running JIRA over HTTPS**

The following flowchart shows the process involved in configuring HTTPS on Tomcat. Click the links below this chart to go to the instructions for that step.



* [Configure HTTPS in Tomcat](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-ConfigureHTTPSInTomcat)
* [Generate Self-Signed Certificate](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-GenerateSelfSignedCertificate)
* [Obtain CA Certificate](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-ObtainCACertificate)
* [Import Certificate into the Trust-store](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-ImportCertificateIntoTheTruststore)
* [Requiring HTTPS for certain pages (Redirecting certain pages to HTTPS)](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-RedirectingCertainPagesToHTTPS)

**Windows Standalone installed using the '**[**Windows Installer**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+Standalone+on+Windows)**':**

"<install\_dir>\jre\bin\keytool" -genkey -alias tomcat -keyalg RSA

**Windows**[**WAR distribution**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+WAR)**and**[**Standalone 'archive' installations**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+Standalone+from+an+Archive+File+on+Windows%2C+Linux+or+Solaris)**:**

"%JAVA\_HOME%\bin\keytool" -genkey -alias tomcat -keyalg RSA

**Unix/Linux**[**WAR distribution**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+WAR)**and**[**Standalone 'archive' installations**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+Standalone+from+an+Archive+File+on+Windows%2C+Linux+or+Solaris)**:**

$JAVA\_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA

This will create (if it doesn't already exist) a new .keystore file located in the home directory of the user you used to run the keytool command.

You will now need to export the certificate to make it ready for importing into the Trust-store with the following command:

**Windows Standalone installed using the '**[**Windows Installer**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+Standalone+on+Windows)**':**

"<install\_dir>\jre\bin\keytool" -export -alias tomcat -file file.cer

**Windows**[**WAR distribution**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+WAR)**and**[**Standalone 'archive' installations**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+Standalone+from+an+Archive+File+on+Windows%2C+Linux+or+Solaris)**:**

"%JAVA\_HOME%\bin\keytool" -export -alias tomcat -file file.cer

**Unix/Linux**[**WAR distribution**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+WAR)**and**[**Standalone 'archive' installations**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+Standalone+from+an+Archive+File+on+Windows%2C+Linux+or+Solaris)**:**

$JAVA\_HOME/bin/keytool -export -alias tomcat -file file.cer

Next, [import the certificate into the Trust-store](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-ImportCertificateIntoTheTruststore).

[^Back to the flowchart](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-RunningJIRAOverHTTPS)

### **Obtain CA Certificate**

Digital Certificate that are issued by trusted 3rd party CAs (Certification Authority) provide verification that your Website does indeed represent your company, thereby verifying your company's identity. Many CAs simply verify the domain name and issue the certificate, whereas other such as [VeriSign](http://www.verisign.com/) verifies the existence of your business, the ownership of your domain name, and your authority to apply for the certificate, providing a higher standard of authentication.

A list of CA's can be found [here](http://www.dmoz.org/Computers/Security/Public_Key_Infrastructure/PKIX/Tools_and_Services/Third_Party_Certificate_Authorities/).  
Some of the most well known CAs are:

* [Verisign](http://www.verisign.com/)
* [Thawte](http://www.thawte.com/)
* [CAcert](http://www.cacert.org/) (relatively new CA, providing free CA certificates)

Next, [import the certificate into the Trust-store](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-ImportCertificateIntoTheTruststore).

[^Back to the flowchart](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-RunningJIRAOverHTTPS)

### **Import Certificate into the Trust-store**

Your SSL Vendor may have different instructions, please refer to them for proper certificate installation. Examples include [GoDaddy](http://help.godaddy.com/article/5239) and [VeriSign](https://knowledge.verisign.com/support/ssl-certificates-support/index?page=content&amp;id=AR234)

 If you want to know where your keystore is located before performing this step, or you receive an error indicating that your keystore cannot be found, please refer to the [Can't find the keystore](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-Cantfindthekeystore) section below.

Assuming your certificate is called "file.cer" whether obtained by a CA or self-generated, the following command will add this certificate to the Trust-store:

**Windows Standalone installed using the '**[**Windows Installer**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+Standalone+on+Windows)**':**

"<install\_dir>\jre\bin\keytool" -import -alias tomcat -file file.cer

**Windows**[**WAR distribution**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+WAR)**and**[**Standalone 'archive' installations**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+Standalone+from+an+Archive+File+on+Windows%2C+Linux+or+Solaris)**:**

"%JAVA\_HOME%\bin\keytool" -import -alias tomcat -file file.cer

**Unix/Linux**[**WAR distribution**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+WAR)**and**[**Standalone 'archive' installations**](https://confluence.atlassian.com/display/JIRA044/Installing+JIRA+Standalone+from+an+Archive+File+on+Windows%2C+Linux+or+Solaris)**:**

This step must be performed as the root user, or with the use of sudo

$JAVA\_HOME/bin/keytool -import -alias tomcat -file file.cer

Next, proceed to the step on [redirecting certain pages to HTTPS](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-RedirectingCertainPagesToHTTPS).

[^Back to the flowchart](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-RunningJIRAOverHTTPS)

### **Configure HTTPS in Tomcat**

Edit conf/server.xml, and at the bottom before the </Service> tag, add this section (or uncomment it where you find it) in Tomcat 6:

|  |
| --- |
| <Connector port="8443" maxHttpHeaderSize="8192" SSLEnabled="true"  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"  enableLookups="false" disableUploadTimeout="true" useBodyEncodingForURI="true"  acceptCount="100" scheme="https" secure="true"  clientAuth="false" sslProtocol="TLS" /> |

This enables SSL access on port 8443 (the default for HTTPS is 443, but just as Tomcat uses 8080 instead of 80 to avoid conflicts, 8443 is used instead of 443 here).

[^Back to the flowchart](https://confluence.atlassian.com/display/JIRA044/Running+JIRA+over+SSL+or+HTTPS#RunningJIRAoverSSLorHTTPS-RunningJIRAOverHTTPS)

**Go to below path where server.xml is there and put below entry:**

**/opt/atlassian/jira/conf**

### 

<Connector port="8085" relaxedPathChars="[]|" relaxedQueryChars="[]|{}^&#x5c;&#x60;&quot;&lt;&gt;" maxHttpHeaderSize="8192" protocol="HTTP/1.1" schema="https" secure="true" SSLEnabled="true" maxThreads="150" minSpareThreads="25" maxSpareThreads="75" clientAuth="false" sslProtocol="TLS" SSLCertificateFile="/opt/atlassian/jira/conf/devopsdev.statebanktimes.in.crt"

SSLCertificateKeyFile="/opt/atlassian/jira/conf/devops.key" SSLVerifyClient="optional" SSLProtocol="TLSv1+TLSv1.1+TLSv1.2" enableLookups="false" disableUploadTimeout="true" useBodyEncodingForURI="true" acceptCount="100" scheme="https" />

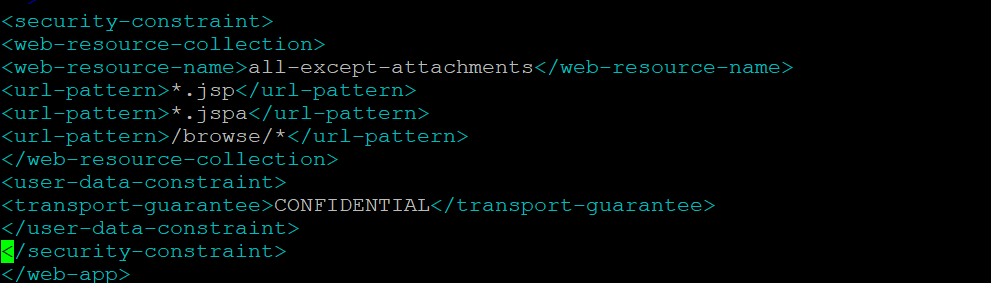
**put this enty in setenv.sh**

export LD\_LIBRARY\_PATH='$LD\_LIBRARY\_PATH:/usr/local/apr/lib'

In the same file we need to put JRE\_HOME AND JAVA\_HOME.

**Go to below path and put that entry:**

/opt/atlassian/jira/atlassian-jira/WEB-INF



### **Redirecting certain pages to HTTPS**

Although HTTPS is now activated and available, the old HTTP URLs ([http://localhost:8080](http://localhost:8080/)) are still available. In most situations one wants these URLs to continue working, but for some to redirect to their https equivalent. This is done by editing WEB-INF/web.xml, and adding the following section at the end of the file, before the closing </web-app>:

|  |
| --- |
| <security-constraint>  <web-resource-collection>  <web-resource-name>all-except-attachments</web-resource-name>  <url-pattern>\*.jsp</url-pattern>  <url-pattern>\*.jspa</url-pattern>  <url-pattern>/browse/\*</url-pattern>  </web-resource-collection>  <user-data-constraint>  <transport-guarantee>CONFIDENTIAL</transport-guarantee>  </user-data-constraint>  </security-constraint> |

This means that all URLs except attachments are redirected from HTTP to HTTPS. IE has a [bug](http://jira.atlassian.com/browse/JRA-8179) which prevents attachments like .doc files being viewed via HTTPS if SSL protection is forced in web.xml.

Once this change is made, restart JIRA and access [http://localhost:8080](http://localhost:8080/). You should be redirected to <https://localhost:8443/secure/Dashboard.jspa>. The port it redirects to is determined by the redirectPort value you specify in the server.xml file in the HTTP Connector stanza.

 There does not seem to be an easy way to make subsequent pages revert to HTTP after logging in via HTTPS - see [JRA-7250](http://jira.atlassian.com/browse/JRA-7250)

## **Troubleshooting**

Here are some troubleshooting tips if you are using a self-signed key created by keytool, as described above.

When you enter "https://localhost:8443" in your browser, if you get a message such as "Cannot establish a connection to the server at localhost:8443", look for error messages in your logs/catalina.out log file. Here are some possible errors with explanations:

### **SSL + Apache + IE problems**

Some people have reported errors when uploading attachments over SSL using IE. This is due to an IE bug, and can be fixed in Apache by setting:

|  |
| --- |
| BrowserMatch ".MSIE." \  nokeepalive ssl-unclean-shutdown \  downgrade-1.0 force-response-1.0 |

[Google](http://www.google.com.au/search?q=BrowserMatch+IE+downgrade-1.0+force-response-1.0&amp;start=0&amp;ie=utf-8&amp;oe=utf-8&amp;client=firefox&amp;rls=org.mozilla:en-US:unofficial) has plenty more on this.

### **Can't find the keystore**

java.io.FileNotFoundException: /home/user/.keystore (No such file or directory)

This indicates that Tomcat cannot find the keystore. The keytool utility creates the keystore as a file called .keystore in the current user's home directory. For Unix/Linux the home directory is likely to be /home/<username>. For Windows it is likely to be C:\Documents And Settings\<UserName>.

Make sure you are running JIRA as the same user who created the keystore. If this is not the case, or if you are running JIRA on Windows as a service, you will need to specify where the keystore file is in conf/server.xml. Add the following attribute to the connector tag you uncommented:

keystoreFile="<location of keystore file>"

### **Incorrect password**

java.io.IOException: Keystore was tampered with, or password was incorrect

You used a different password than "changeit". You must either use "changeit" for both the keystore password and for the key password for Tomcat, or if you want to use a different password, you must specify it using the keystorePass attribute of the Connector tag, as described above.

### **Passwords don't match**

java.io.IOException: Cannot recover key

You specified a different value for the keystore password and the key password for Tomcat. Both passwords must be the same.

### **Wrong certificate**

javax.net.ssl.SSLException: No available certificate corresponds to the SSL cipher suites which are enabled.

If the Keystore has more than one certificate, Tomcat will use the first returned unless otherwise specified in the SSL Connector in conf/server.xml.

Add the keyAlias attribute to the Connector tag you uncommented, with the relevant alias, for example:

<Connector port="8443" maxHttpHeaderSize="8192"

maxThreads="150" minSpareThreads="25" maxSpareThreads="75"

enableLookups="false" disableUploadTimeout="true" useBodyEncodingForURI="true"

acceptCount="100" scheme="https" secure="true"

clientAuth="false" sslProtocol="TLS"

keystoreFile="/opt/local/.keystore"

keystorePass="removed"

keyAlias="tomcat"/>

### **Using Apache Portable Runtime**

APR uses a different SSL engine, and you will see an exception like this in your logs

SEVERE: Failed to initialize connector [Connector[HTTP/1.1-8443]]

LifecycleException: Protocol handler initialization failed: java.lang.Exception: No Certificate file specified or invalid file format

The reason for this is that the APR Connector uses OpenSSL and cannot use the keystore in the same way. You can rectify this in one of two ways:

#### Use the Http11Protocol to handle SSL connections

Edit the server.xml so that the SSL Connector tag you just uncommented specifies the Http11Protocol instead of the APR protocol

<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"

maxHttpHeaderSize="8192" SSLEnabled="true" keystoreFile="${user.home}/.keystore"

maxThreads="150" enableLookups="false" disableUploadTimeout="true"

acceptCount="100" scheme="https" secure="true"

clientAuth="false" sslProtocol="TLS" useBodyEncodingForURI="true"/>

#### Configure the Connector to use the APR protocol

This is only possible if you have PEM encoded certificates and private keys. If you have used OpenSSL to generate your key, then you will have these PEM encoded files - in all other cases contact your certificate provider for assistance.

<Connector

port="8443" maxThreads="200"

scheme="https" secure="true" SSLEnabled="true"

SSLCertificateFile="${user.home}/certificate.pem"

SSLCertificateKeyFile="${user.home}/key.pem"

clientAuth="optional" SSLProtocol="TLSv1"/>

### **Enabling Client Authentication**

To enable client authentication in Tomcat, ensure that the value of the clientAuth attribute in your Connector element of your Tomcat's server.xml file is true.

<Connector

...

clientAuth="true"

... />

For more information about Connector element parameters, please refer to the 'SSL Support' section of the [Tomcat 6.0](http://tomcat.apache.org/tomcat-6.0-doc/config/http.html#SSL_Support) or [Tomcat 5.5](http://tomcat.apache.org/tomcat-5.5-doc/config/http.html#SSL_Support) documentation.

v