Atlassian Knowledge Base

Unable to Connect to SSL Services due to PKIX Path Building Failed

This Knowledge Base article was written specifically for the **Atlassian Server** platform. Due to the Restricted functions in Atlassian Cloud apps, the contents of this article **cannot be applied to Atlassian Cloud applications**.

The content on this page relates to platforms which are not supported. Consequently, Atlassian Support **cannot guarantee providing any support for it**. Please be aware that this material is provided for your information only and using it is done so at your own risk.

Problem

Attempting to access applications that are encrypted with SSL (for example HTTPS, LDAPS, IMAPS) throws an exception and the connection is refused. This can happen when attempting to establish a secure connection to any of the following:

- Active Directory server
- Mail server
- Another Atlassian application using Application Links

For example, the following error appears in the UI when Using the JIRA Issues Macro:

```
Error rendering macro: java.io.IOException: Could not download:
https://siteURL/jira/secure/IssueNavigator.jspa?
view=rss&&type=12&type=4&type=3&pid=10081&resolution=1&fixfor=10348&sorter/field=issuekey
&sorter/order=DESC&sorter/field=priority&sorter/order=DESC&tempMax=100&reset=true&decorat
or=none
```

While the following appears in the logs:

```
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path
building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid
building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid
certification path to requested target
```

Diagnosis

Use SSLPoke to verify connectivity

Try the Java class `SSLPoke` to see if your truststore contains the right certificates. This will let you connect to a SSL service, send a byte of input, and watch the output.

1. Download SSLPoke.class
2. Execute the class as per the below, changing the URL and port appropriately.

```
<JAVA_HOME>/bin/java SSLPoke jira.example.com 443
```

A mail server may be `mail.example.com 465`.

- A failed connection would produce the below:

```
/usr/bin/java SSLPoke jira.example.com 443
sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
    at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:387)
    at sun.security.validator.PKIXValidator.engineValidate(PKIXValidator.java:292)
    at sun.security.validator.Validator.validate(Validator.java:260)
    at sun.security.ssl.X509TrustManagerImpl.validate(X509TrustManagerImpl.java:324)
    at sun.security.ssl.X509TrustManagerImpl.checkTrusted(X509TrustManagerImpl.java:229)
    at sun.security.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.java:124)
    at sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1351)
    at sun.security.ssl.ClientHandshaker.processMessage(ClientHandshaker.java:156)
    at sun.security.ssl.Handshaker.processLoop(Handshaker.java:925)
    at sun.security.ssl.Handshaker.process_record(Handshaker.java:860)
    at sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:1043)
    at
sun.security.ssl.SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:1343)
    at sun.security.ssl.SSLSocketImpl.writeRecord(SSLSocketImpl.java:728)
    at sun.security.ssl.AppOutputStream.write(AppOutputStream.java:123)
    at sun.security.ssl.AppOutputStream.write(AppOutputStream.java:138)
    at SSLPoke.main(SSLPoke.java:31)
Caused by: sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target
    at
sun.security.provider.certpath.SunCertPathBuilder.build(SunCertPathBuilder.java:145)
    at
sun.security.provider.certpath.SunCertPathBuilder.engineBuild(SunCertPathBuilder.java:131)
    at java.security.cert.CertPathBuilder.build(CertPathBuilder.java:280)
    at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:382)
    ... 15 more
```

- A successful connection would look like this:

```
<JAVA_HOME>/bin/java SSLPoke jira.example.com 443
Successfully connected
```

If `-Djavax.net.ssl.trustStore` is present in your JVM arguments, Java will use the keystore specified with that argument. You can verify whether the `-Djavax.net.ssl.trustStore` parameter is causing problems by running the `SSLPoke` test and specifying the same JVM argument to use that keystore. For example:

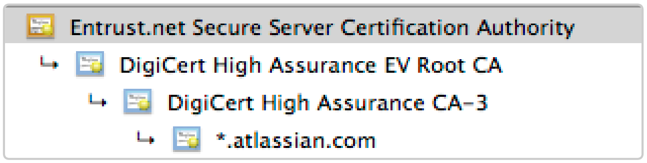
```
<JAVA_HOME>/bin/java -Djavax.net.ssl.trustStore=/my/custom/truststore
SSLPoke jira.example.com 443
```

If this fails (confirming the problem that the truststore doesn't contain the appropriate certificates), then the certificate will need to be imported into that truststore as per the instructions in Connecting to SSL Services.

Cause

Whenever Java attempts to connect to another application over SSL (e.g.: HTTPS, IMAPS, LDAPS), it will *only* be able to connect to that application if it can trust it. The way trust is handled in the Java world is that you have a keystore (typically `$JAVA_HOME/lib/security/cacerts`), also known as the truststore. This contains a list of all known Certificate Authority (CA) certificates, and Java will only trust certificates that are signed by one of those CAs or public certificates that exist within that keystore. For example, if we look at the certificate for Atlassian, we can see that the ***.atlassian.com** certificate has been signed by the intermediate certificates, **DigiCert High Assurance EV Root CA** and **DigiCert High**

Assurance CA-3. These intermediate certificates have been signed by the root **Entrust.net Secure Server CA**:


Entrust.net Secure Server Certification Authority
↳ DigiCert High Assurance EV Root CA
↳ DigiCert High Assurance CA-3
↳ *.atlassian.com

These three certificates combined are referred to as the certificate chain, and, as they are all within the Java keystore (`cacerts`), Java will trust any certificates signed by them (in this case, ***.atlassian.com**). Alternatively, if the ***.atlassian.com** certificate had been in the keystore, Java would also trust that site.

This problem is therefore caused by a certificate that is self-signed (a CA did not sign it) or a certificate chain that does not exist within the Java truststore. Java does not trust the certificate and fails to connect to the application.

Resolution

✓

Add SSL Certificates automatically!

We have SSL for JIRA and SSL for Confluence add-ons available for this process. Please install and use these add-ons for a simpler way to import the certificate.

SSL for JIRA and SSL for Confluence add-ons are not supported by Atlassian.
These plugins are part of Atlassian Labs and may not be compatible with the latest versions of JIRA and Confluence

1. Make sure you have imported the public certificate of the target instance into the truststore according to the Connecting to SSL Services instructions.
2. Make sure any certificates have been imported into the correct truststore; you may have multiple JRE/JDKs. See Installing Java for this.
3. Check to see that the correct truststore is in use. If `-Djavax.net.ssl.trustStore` has been configured, it will override the location of the default truststore, which will need to be checked.
4. Check if your Anti Virus tool has "SSL Scanning" blocking SSL/TLS. If it does, disable this feature or set exceptions for the target addresses (check the product documentation to see if this is possible).
5. If connecting to a mail server, such as Exchange, ensure authentication allows plain text.
6. Verify that the target server is configured to serve SSL correctly. This can be done with the SSL Server Test tool.

Was this helpful? Yes No

Have a question about this article?

Ask our community Ask question

See questions about this article