Deployment Guide

Secure Connections to HTTP Clients

5/5/2025

Secure Connections to HTTP Clients

Contents

The Jetty web server supplied with the Genesys Web Engagement solution includes a pre-configured, self-signed certificate. This allows you to use HTTPS out of the box in a lab deployment.

For a production deployment, you should use a certificate issued by a third-party Certificate Authority. The procedures on this page provide examples of ways to load SSL certificates and configure Jetty. These examples may vary depending on your environment.

Loading an SSL Certificate and Private Key into a JSSE Keystore

> **Important**
> In a development environment, you can use self-signed certificates, but in a production environment you should use a certificate issued by a third-party Certificate Authority, such as VeriSign.

Prerequisites

- An SSL certificate, either generated by you or issued by a third-party Certificate Authority.

Start

1. Depending on your certificate format, do **one** of the following:
    - If your certificate is in PEM form, you can load it to a JSSE keystore with the keytool using the following command:
      ```
      keytool -keystore keystore -importcert -alias alias -file certificate_file
       -trustcacerts
      ```
      Where:

      keystore is the name of your JSSE keystore.

      alias is the unique alias for your certificate in the JSSE keystore.

      certificate_file is the name of your certificate file. For example, `jetty.crt`.
    - If your certificate and key are in separate files, you must combine them into a PKCS12 file before loading it to a keystore.
        1. Use the following command in openssl to combine the files:
           ```
           openssl pkcs12 -inkey private_key -in certificate -export -out pkcs12_file
           ```
           Where:

           private_key is the name of your private key file. For example, `jetty.key`.

           certificate is the name of your certificate file. For example, `jetty.crt`.

           pkcs12_file is the name of the PKCS12 file that will be created. For example, `jetty.pkcs12`.
        2. Load the PKCS12 file into a JSSE keystore using keytool with the following command:

```
keytool -importkeystore -srckeystore pkcs12_file -srcstoretype store_type
 -destkeystore keystore
```

Where:

pkcs12_file is the name of your PKCS12 file. For example, `jetty.pkcs12`.

store_type is the file type you are importing into the keystore. In this case, the type is PKCS12.

keystore is the name of your JSSE keystore.

Important

You will need to set two passwords during this process: keystore and truststore. Make note of these passwords because you will need to add them to your **launcher.ini** configuration file.

End

Next Steps

- Configuring Launcher

Configuring Launcher

Prerequisites

- You completed Loading an SSL Certificate and Private Key into a JSSE Keystore

Start

1. Modify configuration files:
    - Windows
    1. Open the configuration file, ***Web Engagement Root Directory*/server/launcher.ini**, in a text editor.
    2. Find the block of SSL-related parameters, starting from *-Dtrusted-ca*
    3. Fulfill parameters:
        - *-Dtrusted-ca* - Path to trusted CA PEM file or JKS truststore file or SHA-1 Thumbprint for MSCAPI storage.
        - *-Dprovider* - (optional) Type of security provider used. Supported values are PEM, JKS and PKCS11. The provider type will be detected automatically if it is not specified.
        - *-Dtruststore-password* - Password for trust store if trusted CA is in the JKS format.
        - *-Dcertificate-key* - Unencrypted private key in PEM format or Certificate SHA-1 Thumbprint for MSCAPI storage. Ignored for JKS storage.

    - *-Dkeystore-password* - Password for key store if key storage is in the JKS format.
    - *-Dkey-entry-password* - Additional password if the private key is encrypted by its own password.
    - *-Dcertificate* - Client certificate file in PEM format or JKS keystore file or SHA-1 Thumbprint for MSCAPI storage.

- Linux\CentOS

1. Open the configuration file, ***Web Engagement Root Directory*/server/setenv.sh**, in a text editor.
2. Find the block of SSL-related parameters, starting from *#PROVIDER*.
3. Uncomment and fulfill parameters:
    - *TRUSTED_CA* - Path to trusted CA PEM file or JKS truststore file or SHA-1 Thumbprint for MSCAPI storage.
    - *PROVIDER* - (optional) Type of security provider used. Supported values are PEM, JKS and PKCS11. The provider type will be detected automatically if it is not specified.
    - *TRUSTSTORE_PASSWORD' - Password for trust store if trusted CA is in the JKS format.*
    - *PRIVATE_KEY* - Unencrypted private key in PEM format or Certificate SHA-1 Thumbprint for MSCAPI storage. Ignored for JKS storage.
    - *KEYSTORE_PASSWORD* - Password for key store if key storage is in the JKS format.
    - *KEYENTRY_PASSWORD* - Additional password if the private key is encrypted by its own password.
    - *CERTIFICATE* - Client certificate file in PEM format or JKS keystore file or SHA-1 Thumbprint for MSCAPI storage.

- Save your changes.

End

Choosing a Directory for the Keystore

The keystore file in the example above is given relative to the Jetty home directory. For production, you should keep your keystore in a private directory with restricted access. Even though the keystore has a password, the password may be configured into the runtime environment and is vulnerable to theft.

You can now start Jetty the normal way (make sure that **jcert.jar**, **jnet.jar** and **jsse.jar** are on your classpath) and SSL can be used with a URL, such as https://*your_IP*:8743/

Next Steps

- Return to the Genesys Web Engagement Security page.

Choosing a Directory for the Keystore

The keystore file in the example above is given relative to the Jetty home directory. For production, you should keep your keystore in a private directory with restricted access. Even though the keystore has a password, the password may be configured into the runtime environment and is vulnerable to theft.

You can now start Jetty the normal way (make sure that **jcert.jar**, **jnet.jar** and **jsse.jar** are on your classpath) and SSL can be used with a URL, such as https://*your_IP*:8743/

Next Steps

- Return to the Genesys Web Engagement Security page.