



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Genesys Data Processing Server Deployment Guide

Enabling a Secure Connection via HTTPS for GDPS

12/14/2025

Contents

- 1 Enabling a Secure Connection via HTTPS for GDPS
 - 1.1 Process
 - 1.2 Server side
 - 1.3 Client side

Enabling a Secure Connection via HTTPS for GDPS

This article describes how to enable a secure HTTPS connection in GDPS for a Linux environment. There is a detailed discussion on how to configure SSL for Jetty at the following link:
https://wiki.eclipse.org/Jetty/Howto/Configure_SSL

Process

Enabling a secure HTTPS connection in GDPS involves the following steps:

Server side

1. [Creating a secure port.](#)
2. [Creating/importing a certificate.](#)
3. [Updating the Jetty configuration to inject the certificate.](#)

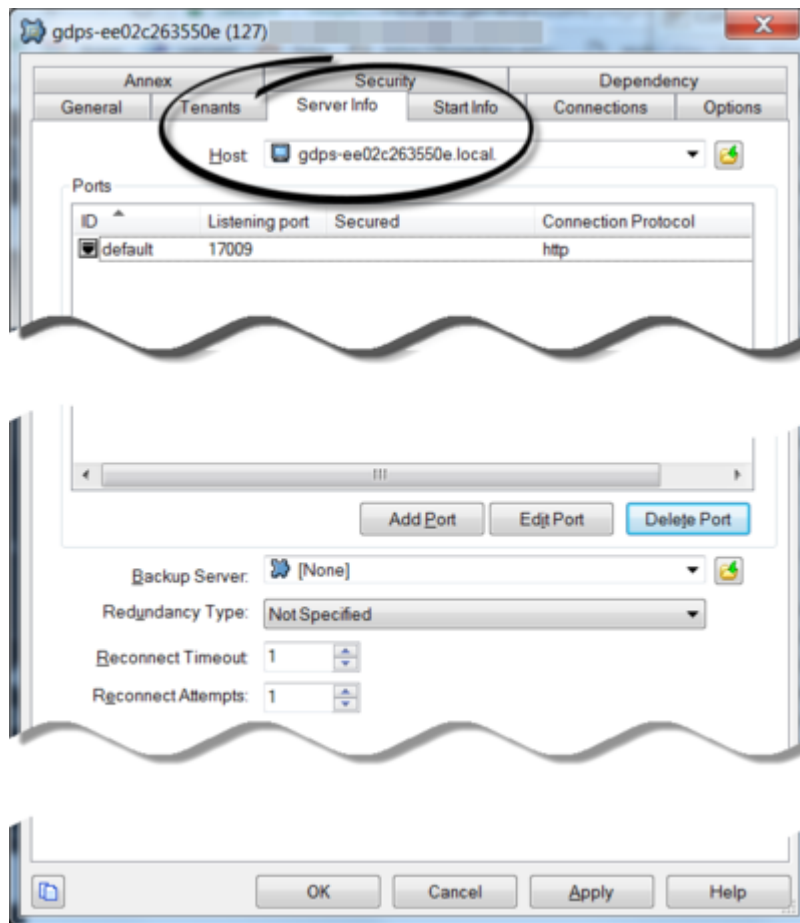
Client side

4. [Confirming/testing with a browser or with curl.](#)
5. [Displaying the certificate.](#)

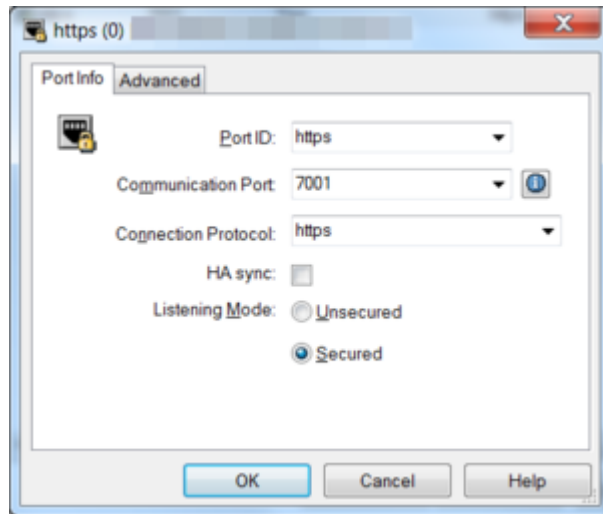
Server side

Creating a secure port

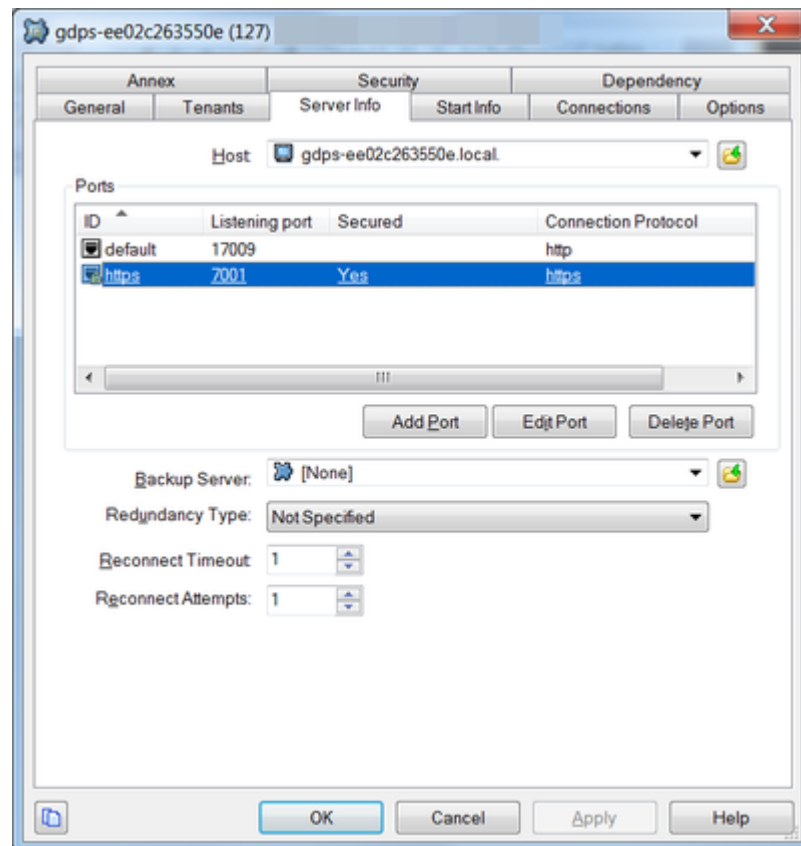
1. Login to Configuration Server and open the GDPS application.
2. Select the **Server Info** tab.



3. Click **Add Port**.



4. Click **Add Port**.
5. Create the new secured port by adding the Port ID https. Note that you can't update the default port from http to https.
6. Set the value of **Communication Port** to a free TCP port.
7. Set the value of **Connection Protocol** to https.
8. For the value of **Listening Mode**, check **Secured**.
9. On the **Advanced** tab, make sure that the **Transport Protocol Parameters** field contains the string `tls=1`.
10. Click OK to save your changes. When saving, you can safely ignore the Listening Mode certificate warning.
11. The configuration should look like this:



Creating/importing a certificate

Creating a certificate

On the GDPS server, you can create a self-signed certificate (for internal or testing purposes) with the Java keytool as follows:

```
keytool -genkey \  
-v \  
-alias gdps2 \  
-dname #:"CN=192.168.99.100,OU=IT,O=JPC,C=GB" \  
-keypass password \  
-keystore gdps2.jks \  
-storepass password \  
-keyalg "RSA" \  
-sigalg "MD5withRSA" \  
-keysize 2048 \  
-validity 365
```

This will create a JKS store named `gdps2.jks`.

Importing a certificate

Import your signed certificate by using the command below. For a production environment, import your signed certificate into a JKS store.

```
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`.

Updating Jetty server to inject certificate

Once you have a certificate (self-signed or imported one), you need to modify Jetty (see the procedure below) so that it fetches your certificate. Jetty provides by default a certificate out of the box. It should be replaced by your certificate.

1. Log into the GDPS server.
2. Navigate to the GDPS install directory.
3. Find the jetty-ssh.xml file. Example: root@ee02c263550e:# cd /gdps/etc root@ee02c263550e:/gdps/etc# ls jetty-ssl.xml jetty-ssl.xml
4. Edit this file, replacing the following lines with the path and password of your certificate:

Original Jetty SSL

```
<Set name="KeyStorePath"><Property name="jetty.base" default="." /></Property name="jetty.keystore" default="etc/keystore"/></Set>
<Set name="KeyStorePassword"><Property name="jetty.keystore.password" default="0BF:1vny1zlo1x8e1vnw1vn61x8g1zlu1vn4"/></Set>
<Set name="KeyManagerPassword"><Property name="jetty.keymanager.password" default="0BF:lu2u1wml1z7slz7a1wnlllu2g"/></Set>
<Set name="TrustStorePath"><Property name="jetty.base" default="." /></Property name="jetty.truststore" default="etc/
keystore"/></Set>
<Set name="TrustStorePassword"><Property name="jetty.truststore.password" default="0BF:1vny1zlo1x8e1vnw1vn61x8g1zlu1vn4"/></Set>
```

New Jetty SSL

```
<Set name="KeyStorePath"><Property name="jetty.base" default="." /></Property name="jetty.keystore" default="gdps2.jks"/></Set>
<Set name="KeyStorePassword"><Property name="jetty.keystore.password" default="password"/></Set>
<Set name="KeyManagerPassword"><Property name="jetty.keymanager.password" default="password"/></Set>
<Set name="TrustStorePath"><Property name="jetty.base" default="." /></Property name="jetty.truststore" default="gdps2.jks"/></Set>
<Set name="TrustStorePassword"><Property name="jetty.truststore.password" default="password"/></Set>
```

Note that the password value should be obfuscated in production mode, using the prefix 0BF: .

5. Save the file.
6. Restart GDPS.

Resources

See documentation on jetty: <https://www.eclipse.org/jetty/documentation/9.3.x/configuring-security-secure-passwords.html>

Example

```
export JETTY_VERSION=9.2.18.v20160721
java -cp lib/jetty-util-$JETTY_VERSION.jar org.eclipse.jetty.util.security.Password root password

2017-11-24 11:12:50.355:INFO::main: Logging initialized @249ms
password
OBF:1v2jluum1xtvlzejlzer1xtnluvk1vlv
MD5:5f4dcc3b5aa765d61d8327deb882cf99
CRYPT:rox7Jdqy.byUU
```

Client side

Confirming/testing with a browser or from the command line

Once GDPS has restarted, test from the client side using either a browser or from the command line.

Browser

From a browser, navigate to the new secured URL—for example, <https://192.168.99.100:7001/data/package/packages>.

Using curl

From the command line, use curl.

1. Make sure that the curl package you are using does support HTTPS by verifying the protocols supported by issuing a `curl --version` command as follows:

```
$ curl --version
curl 7.43.0 (x86_64-w64-mingw32) libcurl/7.43.0 OpenSSL/1.0.2d zlib/1.2.8 libidn/1.32 libssh2/1.6.0 librtmp/2.3
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtmp rtsp scp sftp smtp smtps telnet tftp
Features: IDN Largefile SSPI Kerberos SPNEGO NTLM SSL libz TLS-SRP
```

2. Check that HTTPS is listed. If not, you will get a 'not supported' error message like this one:

```
curl --cacert C:\tmp\ca.crt https://192.168.99.100:7001/data/package/packages
curl: (1) Protocol https not supported or disabled in libcurl
```

3. Once curl is successfully checked, retrieve the certificate by using an openssl command. The public certificate is stored in the ca.crt on the client side. Retrieve the certificate by using a command like this one:

```
openssl s_client -connect 192.168.99.100:7001 -showcerts < /dev/null | openssl x509 -outform PEM > /c/tmp/ca.crt
```

4. Then issue the curl command with the retrieved certificate:

```
curl --cacert /c/tmp/ca.crt https://192.168.99.100:7001/data/package/packages
```

Disabling the certificate check

Alternatively, you can disable certificate check with curl by using the -k option (insecure flag) using a command like this one:

```
curl -k https://192.168.99.100:7001/data/package/packages
```

Displaying the certificate

To display the certificate content from the client, use a command like this one:

```
openssl s_client -connect 192.168.99.100:7001 -showcerts
```

Example:

```
$ openssl s_client -connect 192.168.99.100:7001 -showcerts
Loading 'screen' into random state - done
CONNECTED(00000134)
depth=0 C = GB, O = JPC, OU = IT, CN = 192.168.99.100
verify error:num=18:self signed certificate
verify return:1
depth=0 C = GB, O = JPC, OU = IT, CN = 192.168.99.100
verify return:1
---
Certificate chain
0 s:/C=GB/O=JPC/OU=IT/CN=192.168.99.100
i:/C=GB/O=JPC/OU=IT/CN=192.168.99.100
```

```
-----BEGIN CERTIFICATE-----
MIIDITCCAgmgAwIBAgIESfSLVTANBgkqhkiG9w0BAQQFADBBMQswCQYDVQQGEwJH
QjEMMAoGA1UEChMDSlBDMQswCQYDVQQLEwJJVDEEXMBUGA1UEAxMOMTkyLjE2OC45
OS4xMDAwHhcNMTcxMTI0MDgyNTM5WhcNMjcMTIyMDgyNTM5WjBBMQswCQYDVQQG
EwJHQjEMMAoGA1UEChMDSlBDMQswCQYDVQQLEwJJVDEEXMBUGA1UEAxMOMTkyLjE2
OC45OS4xMDAwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCZiNgEWisR
jGbMDfGLC52dMYblQtC305J0kkqTz675Cc/6AIa/i2KrxJ33nUYb8T9D8b9Y66rt
vzdAZfhirUE9AlxMiIyQqMNa+PahxhpgaYK26u/ev2YGI3EGrQfDXTl1Nbp7pqaS
LaGLK5Dmpo21Ef0sfVnYa8VTc4uI3Q0kqhp2l0MdiHXDvgp6peursRoe6YVbAoV0
acrc19PRFcN6Iir+32wsSj0f8DlxbypfbQf1BK4V26SyNW5DD4gEajPfVQfuh+s
d0umlegvvv00j60HNPSFxJLWPsG2x+3L+9pA+WsmJ8DSRktCimsdH69V9jNKp36k
Kjz1CMCQogrvAgMBAAGjITAFMB0GA1UdDgQWBBQMqJkbaW5STz1JrNb9jh6ySlQy
bjANBgkqhkiG9w0BAQQFAA0CAQEAAaEAKdMEd0P9yC1o3cWkKXWTs0aQduGKV4K1x
drCGTnwzPG2bsz0z4VfZwpHjKlBv+Yeo+FR/Bz6Y/ByBmj jkGEAKXwoHnon8qcs+
xEc9l79c904vbB6W0W3BG5LmoyyXeYuOpJ0qHFUVpHmrz8sDwK57F53kASomH0Y1
5daTXWSt/XHyNccaJdRTf1PatzE7fo/tTw2l8jvQzBycPXi88fV2gTiTXst2Jzxq
Hvp+JqkRc9P2KMq3TW3lmVJ2jHNyegrNAsfHFh/oHWu4Z0xxDD25KtWrR7Nxu+p
6BXNI2E4ZvHiMCYwna+ThjgT+0aH+HhXcRtfQX/Hjkl+aqdoPQ==
-----END CERTIFICATE-----
---
Server certificate
subject=/C=GB/O=JPC/OU=IT/CN=192.168.99.100
issuer=/C=GB/O=JPC/OU=IT/CN=192.168.99.100
---
No client certificate CA names sent
Peer signing digest: SHA512
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 1289 bytes and written 444 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
Protocol : TLSv1.2
Cipher : ECDHE-RSA-AES128-GCM-SHA256
Session-ID: 5A17F573EBE27C68EBD5213F42349E364EBDC39325D36E59D0BAF9965BC4905C
Session-ID-ctx:
Master-Key: B353A0A3A8158CE021F9ED2516AB8422D410DBF8EFA4D6AC8445C07AF7A1B14AAEF8BC6E3BB677EA7FDEFEA2048A07EE
Key-Arg : None
```

```
PSK identity: None
PSK identity hint: None
SRP username: None
Start Time: 1511519603
Timeout : 300 (sec)
Verify return code: 18 (self signed certificate)
---
closed
```

Other resources

- <https://www.eclipse.org/jetty/documentation/9.3.x/configuring-security-secure-passwords.html>
- Download a curl for Windows that supports HTTPS from this location: <https://curl.haxx.se/download.html>