Securing your Acrolinx Server with HTTPS

support.acrolinx.com/hc/en-us/articles/213614069

Applies To

Software	Version
Acrolinx Server	5.0, 5.1

In a standard installation of Acrolinx, communication between the server and the outside world is unencrypted. By "outside world" we mean people who connect to the server with an Acrolinx plug-in, or with a web browser.

Unencrypted communication is normally not an issue when the server and clients both run within the same corporate network. When the clients are connecting through the public internet, however, you might choose to use HTTPS, the internet standard for secure web communication. HTTPS communication ensures that data and passwords are unreadable during transmission.

Generally, we recommend that you always configure a cipher whitelist and disable unused and insecure SSL protocols . We've added the recommended configurations in the respective sections.

By the way, setting up HTTPS, especially generating the proper public key certificate, is best done by a security professional. Some of the configuration steps aren’t for the faint-hearted.

Trusting a Third-Party Certificate

A third-party certificate could be a certificate that you received from a certificate authority to secure your Acrolinx server. Alternatively, a third-party certificate could also be a certificate from a third-party server that the Acrolinx server interacts with. The most common example is a server that hosts an LDAP directory service. In both cases, you must configure the Acrolinx server to trust the certificate before the server can communicate securely.

The are two methods for trusting a third-party certificate. The easiest method is to move the certificate to the server's "trusted" directory. If that method doesn't work, you can also use the Java keytool to import the certificate into the Java keystore.

Moving the Third-Party Certificate to the Trusted Directory

The preferred method to trust a third-party certificate is to move it to the server's "trusted" directory.

To trust an untrusted certificate, follow these steps:

1. Follow this step if you're connecting to third-party server such as an LDAP server.

   Open the following directory and copy the certificate that the Acrolinx server downloaded from your third-party server.

   `%ACROLINX_CONFIGURATION_ROOT%\server\certificates\untrusted\`

2. Place your third-party certificate to the following directory:

   `%ACROLINX_CONFIGURATION_ROOT%\server\certificates\trusted\`

   Your changes take effect immediately and the server successfully authenticates users on the next connection attempt.

Trusting a Third-Party Certificate with the Java Keytool

To secure the Acrolinx Server for HTTPS communication, you must define the protocol and port that you use for secure communication. Additionally, you must define the keystore and truststore details.

To configure the HTTPS properties for the core server, follow these steps:

1. Import your certificate into a Java keystore.

   a. If the certificate is in `<CERTIFICATE_NAME>.crt` and the key is in `<KEY_NAME>.key` , then first combine the files into a single PKCS12 file.

      Open a command-line shell and enter the following command:

      ```
      openssl pkcs12 -inkey <KEY_NAME>.key -in <CERTIFICATE_NAME>.crt
      -export -out combined.pkcs12
      ```

      NOTE: It’s possible that some browsers display a warning that the certificate isn’t trustworthy. In that case, you should combine *all* certificate files you received together with the `<CERTIFICATE_NAME>.crt` file into a new `<CERTIFICATE_NAME>.crt` file and run the command again. The PKCS12 file now includes the complete keychain and all browsers should trust the certificate.

   b. Enter the `<PASSWORD>` when asked.

   c. Copy the file `combined.pkcs12` from wherever it was generated into the `bin` directory of your server.

      For example, you might enter a command that resembles the following example:

      ```
      copy  C:\files\combined.pkcs12
      C:\Acrolinx\server\bin
      ```

   d. Import the PKCS12 file into your Java keystore using the following command, run from your server's `bin` directory.

      For example, you might first enter commands that resemble the following example:

```
cd C:\Acrolinx\server\bin
java -jar "../lib/jetty-pkcs12-<VERSION_NUMBER>.jar"
combined.pkcs12 myKeystore
```

The variable `<VERSION_NUMBER>` indicates the version number of the currently installed PKCS12 importer jar.

To locate the version number, check the directory `<INSTALL_DIR>\server\lib` for a file name that matches the pattern `jetty-pkcs12-<VERSION_NUMBER>.jar` .

For example, your final commands might look like this:

```
cd C:\Acrolinx\server\bin
java -jar "../lib/jetty-pkcs12-jetty-pkcs12-8.1.16.v20140903.jar"
combined.pkcs12 myKeystore
```

This will generate a keystore in your server's `bin` directory.

2. Open your overlay of the core server properties file and add the following properties.

   ```
   endpointHost=<SERVER_NAME>
   endpointProtocol=https
   ```

   Set `endpointHost` to the name of the server and not to the IP address like this: `endpointHost=acroserver` . To use HTTPS, the core server must have a DNS name that is accessible to the clients.

   Update the `endpointPort` value if you use a different port for secure communication.

   By default, all clients and language servers connect to the core server through port `8031` . If your security policy requires secure communication to go through a specific port such as `443` , add the extra `endpointPort` as follows:

   ```
   endpointPort=443
   ```

   You find the overlay for the core server properties file in the following location:

   ```
   %ACROLINX_CONFIGURATION_ROOT%\server\bin\coreserver.properties
   ```

3. Define the keystore and truststore details by adding the following properties:

   ```
   sslKeystore=<KEYSTORE_NAME>
   sslKeystorePassword=<PASSWORD>
   sslTruststore=<KEYSTORE_NAME>
   sslTruststorePassword=<PASSWORD>
   ```

   Example:

   ```
   sslKeystore=myKeystore
   sslKeystorePassword=pRu4Hatr
   sslTruststore=myKeystore
   sslTruststorePassword=pRu4Hatr
   ```

   In most cases, the value for `sslTruststore` is the same as the value for `sslKeystore` . If you omit the `sslTruststore` properties completely, the values that you entered for the `sslKeystore` properties are automatically used as the `sslTruststore` settings as well.

   If you created a separate truststore, use that file name and password instead.

4. If you use a different JVM to the one that is installed with the Acrolinx Server and want to specify a different keystore algorithm, add the following additional property:

   ```
   ssl.KeyManagerFactory.algorithm=
   <ALGORITHM_NAME>
   ```

   For example, if you use the IBM JVM add the property as follows:

   ```
   ssl.KeyManagerFactory.algorithm=IbmX509
   ```

5. Save your changes and restart the core server.

If you want to be extra secure, configure the language servers to communicate over HTTPS.

You don't have to do this, but it's an extra precaution to prevent attacks from within your network. You might also secure internal communication if your language server and core server are both accessible over the internet.
When you secure the communication between the language servers and the core server, they use the Transport Layer Security (TLS) protocol by default. This is the latest and most secure protocol.

To configure the language servers to communicate over HTTPS, follow these steps:

a. Open your overlay of each language server properties file.

   You find the overlays for the language server properties files in the following location:

   ```
   %ACROLINX_CONFIGURATION_ROOT%\server\bin\ls-
   <NUMBER>.properties
   ```

   The placeholder `<NUMBER>` indicates the number of each language server in the format `ls-01` , `ls-02` , and so on.

b. In each file, update the properties according to the following example:

   ```
   coreServerEndpointUrl=https://<HOST_NAME>:<HOST_PORT>/internal/core
   endpointHost=<HOST_NAME>
   endpointProtocol=https
   ```

   Replace the placeholder `<HOST_NAME>` with the name of the computer where the core server is running. It's important that you use the host name rather than the IP address, otherwise the configuration might not work correctly.

For example, if your server computer is called "acroserver" and you use the default port `8031` , you would update the properties as follows:

```
coreServerEndpointUrl=https://acroserver:8031/internal/core
endpointHost=acroserver
endpointProtocol=https
```

c. Save your changes and restart the language server.

When you restart the language server, it will automatically create a certificate and attempt to connect to the core server. The core server initially rejects this communication because it doesn't recognize the generated certificate. However, there's no need for alarm, just proceed to the next step.

d. Import the generated certificate into the Java keystore for your language server.

Your server certificate is stored in the following directory:

`%ACROLINX_CONFIGURATION_ROOT%/server/certificates/own/` Import the certificate with the Java keytool. You'll find this tool in the following directory:

`<JAVA_HOME>\bin\keytool.exe`

- If you use Windows, the path to your keytool might resemble the following example: `C:\Program Files\Java\jre8\bin\keytool.exe`
- If you use a Unix-based operating system, the path to your keytool might resemble the following example: `/usr/java/default/bin/keytool`

Use the following commands to import the certificate:

```
cd <INSTALL_DIR>\bin\
<JAVA_HOME>\bin\keytool -importcert -file %ACROLINX_CONFIGURATION_ROOT%/server/certificates/own/<CERTIFICATE_NAME> -keystore
myKeystore -alias ls-<NUMBER>
```

For example, if your certificate for language server `01` is called `CN\=acroserver\,DC\=ls-01\,DC\=smarttech\,DC\=com.der` and you run the Acrolinx server on Ubuntu, your commands might resemble the following example:

```
cd /home/admin/Acrolinx/server/bin
/usr/java/default/bin/keytool -importcert -file
/home/admin/.config/Acrolinx/ServerConfiguration4.5.0/server/certificates/own/CN\=acroserver\,DC\=ls-
01\,DC\=smarttech\,DC\=com.der -keystore myKeystore -alias ls-01
```

e. Restart the core server.

The core server should now recognize the secure connection from the language server.

Updating the Server Address in the Acrolinx Plug-ins and Clients

After you have secured the Acrolinx Server communication with HTTPS, Acrolinx plug-in users must update the connection details in their plug-ins and clients to use the new HTTPS address.

Instruct plug-in users to navigate to Options > Change Server... and update the **Server address** field to:

`https://<SERVER_NAME>:8031`

NOTE: In some older plug-in versions and integrations, this address format might not work. If users receive an error message after entering this address, try the following address instead:

`https://<SERVER_NAME>:8031/acrolinx/services/core?wsdl`

Users must access the Dashboard through the https address as well:

`https://<SERVER_NAME>:8031/Dashboard.html`

Configuring a Cipher Suite Whitelist

A cipher suite whitelist ensures that the Acrolinx Server only uses certain cipher suites when serving pages over a secure connection. It’s not compulsory to configure a cipher whitelist, but you might configure a whitelist if your security policy only allows highly secure cipher suites when communicating over a secure connection.

To configure a cipher suite whitelist, follow these steps:

1. Open your overlay of the core server properties file.

   You find the overlay for the core server properties file in the following location:

   `%ACROLINX_CONFIGURATION_ROOT%\server\bin\coreserver.properties`

2. Add the following property:
   `sslCipherSuiteWhitelist=<CIPHER_SUITE_NAMES>`

   Enter the cipher suite names in a comma-separated list. We recommend that you allow only highly secure cipher suites and exclude medium and weak cipher suites. For this setup, you would enter the property as follows:

Example:

```
sslCipherSuiteWhitelist=TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
```

3. Save your changes and restart the core server.

Disabling SSL Protocols

For maximum security, you should only use secure SSL protocols. Currently, this means using the TLS protocol, preferably in version 1.2. In Acrolinx, you can disable individual SSL protocols that are either no longer secure or might be insecure in certain configurations.

When disabling SSL protocols, you should always take a detailed inventory of your currently used integrations. Some older integrations - especially custom integrations - might not work with the new and secure protocols.

To disable SSL protocols, follow these steps:

1. Open your overlay of the core server properties file.

   You find the overlay for the core server properties file in the following location:

   ```
   %ACROLINX_CONFIGURATION_ROOT%\server\bin\coreserver.properties
   ```

2. Add the following property:

   ```
   sslDisabledProtocols=
   <PROTOCOL_NAMES>
   ```

   Enter the SSL protocol names in a comma-separated list. We recommend that you disable all SSL protocols and use only TLS for maximum security. For this setup, you would enter the property as follows:

   ```
   sslDisabledProtocols=SSLv2,SSLv3
   ```

3. Save your changes and restart the core server.