Web Services and Applications Deployment Guide

HTTPS for Elasticsearch

5/3/2025

Contents

HTTPS for Elasticsearch

The HTTPS communication secures the interaction between Elasticsearch and various clients such as web browsers, Postman, and Spring Boot client applications. This requires configuration in both Server and Client.

Server configuration

In Genesys Web Services (GWS) 8.6, you can use your custom Java KeyStore (JKS) file to establish the HTTPS connection between GWS and Elasticsearch.

To establish the HTTPS connection,

1. Copy the **jksStorage.jks** file to the Elasticsearch configuration path: **/usr/share/elasticsearch/config/jksStorage.jks**
2. Enable SSL by configuring the JKS file by using the following settings in the Elasticsearch configuration.

   ```
   xpack.security.enabled: true
    xpack.security.http.ssl.enabled: true
    xpack.security.http.ssl.keystore.type: JKS
    xpack.security.http.ssl.keystore.path: jksStorage.jks
    xpack.security.http.ssl.keystore.password: *******
   ```

   For more details, refer to the Elasticsearch official documentation site.

Important

Once SSL is enabled on the server side, clients must use the same certificate(s) from the JKS file for encrypted connections.

Client (GWS 8.6) configuration

The client side configuration involves configuring the certificate in the client's trustStore. You can do this in two ways:

1. Custom trustStore configuration
2. System default configuration

Custom trustStore configuration

In this method, GWS 8.6 serves as the client.

To enable custom trustStore configuration in GWS 8.6,

1. Copy the JKS file to the client (GWS) machine.
2. Add the following configuration in the **application.yaml** file of the GWS 8.6 application.

```
serverSettings:
  caCertificate: /path/to/jksStorage.jks
  jksPassword: *******

  elasticSearchSettings:
    transportClient:
      nodes:
        - host: 127.0.0.1
          port: 9200
      username: elastic
      password: password
      useTls: true # Enable this for ES https connection
```

Peer verification configuration

You can verify the peer's certificate of other SSL/TLS connections to ElasticSearch for enhanced security by adding the `verifyPeer` option. Add this option in the **elasticSearchSettings** section in the **application.yaml** file of the GWS 8.6 application. By default, this option is set to `true`. For example,

```
elasticSearchSettings:
  verifyPeer: true/false
```

System default configuration

If a custom configuration is not found (that is, if **serverSettings.caCertificate** is not configured), then the system default configuration is used for the client side configuration.

The JDK ships with a limited number of trusted root certificates in the **<java-home>/lib/security/cacerts** file. It is your responsibility to maintain (that is, add/remove) the certificates contained in the default truststore. Depending on the certificate setup of the ElasticSearch server, additional root certificate(s) must be added. For more details, refer the JSSE Reference guide.

From the Elasticsearch server setup in your environment, extract the certificate details from the **.jks** file, and append it to the **<java-home>/lib/security/cacerts** file. The example JKS file name used in this article is **jksStorage.jks**.

To update the **cacerts** file,

1. Export the certificate from the **.jks** file by running the following command:
   ```
   keytool -exportcert -alias your_alias_name -file ca_cert.pem -keystore
    jksStorage.jks
   ```
   If prompted to enter a password, use the password associated with your JKS file.
2. Import the certificate into the Java cacerts file.
   ```
   keytool -importcert -alias your_alias_name -file ca_cert.pem -keystore cacerts
   ```

If prompted to enter a password, use the cacerts password, which is `changeit`.

Enabling anonymous access

When **xpack.security.enabled** is set to `true`, login credentials are required to access Elasticsearch. If you want to set up Elasticsearch without login credentials, you can enable anonymous access.

To enable anonymous access, add the following settings to the Elasticsearch configuration.

```
xpack.security.authc.anonymous.roles: superuser
xpack.security.authc.anonymous.authz_exception: true
```

For details on built-in roles, refer to the Elasticsearch Built-in Roles Documentation.

Elasticsearch connection configuration

Elasticsearch supports the following connections depending on the configured settings:

- HTTP without authentication
- HTTP with authentication
- HTTPS with authentication
- HTTPS without authentication

Settings	HTTP without Authentication	HTTP with Authentication	HTTPS with Authentication	HTTPS without Authentication
xpack.security.enabled	false	true	true	true
xpack.security.http.ssl.enabled	false	false	true	true
ELASTIC_PASSWORD	NA	password	password	password
xpack.security.http.ssl.keystore.type	NA	NA	JKS	JKS
xpack.security.http.ssl.keystore.path	NA	NA	jksStorage.jks	jksStorage.jks
xpack.security.http.ssl.keystore.password	NA	NA	genesys	genesys
xpack.security.authc.anonymous.authz_exception	NA	NA	NA	true
xpack.security.authc.anonymous.roles	NA	NA	NA	superuser

mTLS configuration

To configure a specific keystore/truststore path along with its key alias and password for using mutual TLS (mTLS), add the following option:

```
elasticSearchSettings:
  transportClient:
    keyStorePath: /path/to/jksStorage.jks
    keyStorePassword: password
    keyAlias: key_alias
    keyPassword: password
```

Next Step

- Back to Configuring security