Genesys Mobile Services Deployment Guide

Cassandra Security

12/17/2025

Contents

Cassandra Security

This page discusses security configurations for Cassandra.

Protecting Data Stored in the Cassandra Database

The *<Genesys Mobile Services installation directory>/etc/cassandra.yaml* file enables or disables encryption of Cassandra inter-node communication using `TLS_RSA_WITH_AES_128_CBC_SHA` as the cipher suite for authentication, key exchange, and encryption of the actual data transfers. To encrypt all inter-node communications, set to *all*. You must also generate keys, and provide the appropriate key and trust store locations and passwords. Details about Cassandra options are available from:

- internode_encryption
- authenticator

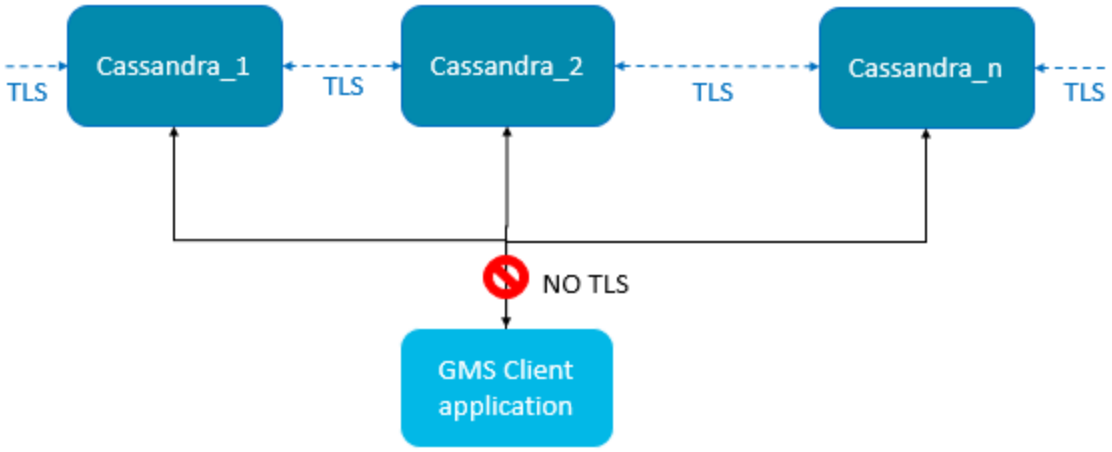
All transient service session-related data is stored in a Cassandra database that uses memory and the file system. See the *<Genesys Mobile Services installation directory>/data* folder. Files located here should be protected from unauthorized access.

Cassandra Authentication

The Cassandra API for custom authentication and authorization has been deprecated in 2.x. Although the legacy classes for authentication and authorization are still implemented in GMS for backward compatibility, Genesys recommends that you use the External Cassandra configuration for both authentication and authorization.

Cassandra TLS Support

Modified in: 8.5.204.00



TLS support for external Cassandra includes:

- Inter-node communication (Gossip)
- JMX connection (Cassandra tools for monitoring)

Cassandra TLS Configuration

Create and Import your Certificate

To enable Cassandra TLS feature, use the JDK `keytool` command to create a certificate per node. For example, the commands below create the keystore.node1 file, then export it as node1.cer, and create the `truststore.node1` file.

```
<GMS client side> $ keytool -genkey -keyalg RSA -alias node1 -keystore keystore.node1
-storepass cassandra -keypass cassandra -validity 36500 -dname "CN=192.168.2.1, OU=None,
O=None, L=None, C=None"
 <GMS client side> $ keytool -export -alias node1 -file node1.cer -keystore keystore.node1 #
password: cassandra
 <GMS client side> $ keytool -import -v -trustcacerts -alias node1 -file node1.cer -keystore
truststore.node1 # new password: cassandra + yes
```

Prepare the keystore file used for cassandra configuration file (cassandra.yaml) by copying the keystore file to in the `<cassandra home>/conf` directory:

```
$ cp keystore.node1 .keystore     ## for Cassandra server side (client_encryption config)
# only for your eyes, restrict access to this file, Keep the crackers from your door
$ chmod 600 .keystore
```

Edit cassandra.yaml to include this file in the Cassandra configuration.

```
native_transport_port_ssl: 9142 # uncomment this line to have both ports (secured on 9142 and
default unsecured on 9042)
# client_encryption_options:
 enabled: true
 keystore: conf/.keystore # keystore file location
```

```
 keystore_password: cassandra # password of keystore
```

Edit launcher.xml

Find the launcher.xml file that is part of the GMS Installation directory. Add these parameters to enable TLS support:

```
<parameter name="cassandranodes_truststore" displayName="cassandratrustore" mandatory="true"
hidden="true" readOnly="true">
     <description><![CDATA[Certificates trustStore for Cassandra nodes]]></description>
     <valid-description><![CDATA[]]></valid-description>
     <effective-description/>
     <format type="string" default="-Djavax.net.ssl.trustStore=client.truststore" />
     <validation></validation>
</parameter>
<parameter name="cassandranodes_trustStorePassword" displayName="cassandratrustStorePassword"
mandatory="true" hidden="true" readOnly="true">
     <description><![CDATA[Certificates trustStore password for Cassandra
nodes]]></description>
     <valid-description><![CDATA[]]></valid-description>
     <effective-description/>
     <format type="string" default="-Djavax.net.ssl.trustStorePassword=cassandra" />
     <validation></validation>
</parameter>
<parameter name="debugtls" displayName="Debug TLS" mandatory="true" hidden="true"
readOnly="true">
     <description><![CDATA[Add debug mode for SSL]]></description>
     <valid-description><![CDATA[]]></valid-description>
     <effective-description/>
     <format type="string" default="-Djavax.net.debug=ssl" />
     <validation></validation>
</parameter>
```

For Windows, Global Certificate File

For Windows, add the Certificates trustStore for Cassandra nodes to the main JRE lib/security/cacerts file:

```
$ keytool -importcert -file cassandranodes.cer -alias cassandranodes -keystore $JAVA_HOME/jre/
lib/security/cacerts -storepass changeit
```

> **Important**
>
> `changeit` is the default password at Java installation time.

Edit the GMS Node Configuration

Here are the cassandra options for TLS cassandra (native-port and secured options) in GMS application options:

```
[cassandra]
native-port=9142
nodes=<cassandra nodes comma separated>
secured=true
strategy-class=SimpleStrategy
```

```
strategy-option=replication_factor:2
```

For further details about these options, refer to the cassandra section of the Options' reference guide.

Cassandra Gossip TLS

In Cassandra version 1.1.x (the Cassandra version in GMS), the internode (gossip) encryption is set up in the *cassandra.yaml* file. Locate the following lines in the *cassandra.yaml* file:

```
encryption_options:
 internode_encryption: none
 keystore: conf/.keystore
 keystore_password: cassandra
 truststore: conf/.truststore
 truststore_password: cassandra
```

Replace `internode_encryption: none` with `internode_encryption: all`, as shown in the following example:

```
encryption_options:
 internode_encryption: all
 keystore: conf/.keystore
 keystore_password: cassandra
 truststore: conf/.truststore
 truststore_password: cassandra
```

For managing keystore and truststore (and password), see the Oracle documentation keytool-Key and Certificate Management Tool or the Oracle security guide.

Cassandra JMX TLS

> ### Warning
> The instructions detailed in this section apply only if Cassandra is embedded in GMS (before GMS version 8.5.203).

Cassandra monitoring and management can be done using a Java Management Extensions (JMX) tool. The JMX access must be protected in order to avoid any remote managing on the GMS embedded Cassandra. To protect JMX access, edit the `launcher.xml` file that contains the following lines (by default):

```
<parameter name="jmxport" displayName="jmxport" mandatory="true"
hidden="true" readOnly="true">
     <description><![CDATA[JMX related]]></description>
     <valid-description><![CDATA[]]></valid-description>
     <effective-description/>
     <format type="string"
default="-Dcom.sun.management.jmxremote.port=9192" />
     <validation></validation>
```

```
 </parameter>
 <parameter name="jmxssl" displayName="jmxssl" mandatory="true"
 hidden="true" readOnly="true">
     <description><![CDATA[virtual machine related]]></description>
     <valid-description><![CDATA[]]></valid-description>
     <effective-description/>
     <format type="string" default="-Dcom.sun.management.jmxremote.ssl=false" />
     <validation></validation>
 </parameter>
 <parameter name="jmxauthenticate" displayName="jmxauthenticate"
 mandatory="true" hidden="true" readOnly="true">
     <description><![CDATA[virtual machine related]]></description>
     <valid-description><![CDATA[]]></valid-description>
     <effective-description/>
     <format type="string"
default="-Dcom.sun.management.jmxremote.authenticate=false" />
     <validation></validation>
 </parameter>
```

By default, the TLS and authentication parameters are disabled:

```
   com.sun.management.jmxremote.ssl=false
   com.sun.management.jmxremote.authenticate=false
```

For information about enabling these parameters and managing JMX and TLS, see the *Monitoring and Management Using JMX Technology* chapter in the Oracle Java SE Monitoring and Management Guide.