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 Driver for SMS and MMS Guide

Security and Privacy

2/13/2025

Security and Privacy

This page contains the following information on configuring security and privacy settings for Genesys Driver for SMS and MMS.

- Configuring security
- Configuring privacy

Configuring security

By default, the driver connects to the SMSC (Short Message Service Center) without any security. However, the driver can use TLS (Transport Layer Security) to connect to the SMSC.

The following section documents how to set up a host certificate.

Configuring the TLS connection

You must add or edit the following options at the channel level:

- x-smpp-tls-mode

> **Important**
>
> The SMSC and the driver must use the same x-smpp-tls-mode setting.

- x-smpp-tls-version
- x-smpp-tls-path-to-certificate
- x-smpp-tls-certificate-file-password
- x-smpp-tls-certificate-password

> **Important**
>
> The options **x-smpp-tls-certificate-file-password** and **x-smpp-tls-certificate-password** are not protected or encrypted. You can hide these options in the driver's log files (see the driver administration page for more information), but they are visible in Genesys administration tools such as Genesys Administrator or Genesys Administrator Extension (GAX).

Use cases

TLS mode 'single'

- Default Java key store is used (file is in **<JRE_folder>\lib\security\cacerts**):
  - x-smpp-tls-mode = single
  - x-smpp-tls-version = Set as needed
- Your private key store is used (only certificates from this key store are available for the driver):
  - x-smpp-tls-mode = single
  - x-smpp-tls-version = Set as needed
  - x-smpp-tls-path-to-certificate = Set to the location of the PKCS#12 file that contains the SMSC’s host certificate with related certificate chains
  - x-smpp-tls-certificate-file-password
  - x-smpp-tls-certificate-password

TLS mode 'mutual'

- Your private key store is used (only certificates from this key store are available for the driver):
  - x-smpp-tls-mode = mutual
  - x-smpp-tls-version = Set as needed
  - x-smpp-tls-path-to-certificate = Set to the location of the PKCS#12 file that contains the SMSC’s host certificate with related certificate chains and private key of the driver's certificate
  - x-smpp-tls-certificate-file-password
  - x-smpp-tls-certificate-password

Possible problems

This section documents some common problems that you might encounter and samples of accompanying log errors.

Certificate file is not trusted by SMSC

```
17:20:06.852 Std 43009 (channel-opensmpp).(oSmpp.openAndBind): error... failed to send bind request:
sun.security.validator.ValidatorException: No trusted certificate found
17:20:06.854 Std 43011 (channel-opensmpp).(oSmpp.openAndBind): exception caught and processed...
*** Start of Trace ***
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: No trusted certificate found
        at sun.security.ssl.Alerts.getSSLException(Alerts.java:192)
        at sun.security.ssl.SSLSocketImpl.fatal(SSLSocketImpl.java:1949)
        at sun.security.ssl.Handshaker.fatalSE(Handshaker.java:302)
```

SMSC is not trusted certificate for your certificate:

```
17:24:36.800 Std 43009 (channel-opensmpp).(oSmpp.openAndBind): error... SMSC close socket connection. Maybe SMSC don't trust your
certificate

17:24:36.802 Std 43011 (channel-opensmpp).(oSmpp.openAndBind): exception caught and processed...
*** Start of Trace ***
java.lang.Exception: SMSC close socket connection. Maybe SMSC don't trust your certificate
        at com.genesyslab.mcr.smserver.channel.sms_mms.opensmpp.SmppDriver.handleExceptionForTLS(SmppDriver.java:612)
        at com.genesyslab.mcr.smserver.channel.sms_mms.opensmpp.SmppDriver.openAndBind(SmppDriver.java:450)
```

Password for certificate file is invalid

```
17:26:23.207 Std 43010 (channel-opensmpp).(oSmpp.initialize): error... failed to initialize ssl context: keystore password was incorrect
17:26:23.208 Std 42403 (PU channel-opensmpp).(<init>): exception caught and processed...
*** Start of Trace ***
java.lang.Exception: channel driver initiaization failed
        at com.genesyslab.mcr.smserver.channel.sms_mms.opensmpp.SmppDriver.initialize(SmppDriver.java:178)
        at com.genesyslab.mcr.smserver.channel.ChannelPU.<init>(ChannelPU.java:123)
        at com.genesyslab.mcr.smserver.channel.ChannelConnectors.processChannelSection(ChannelConnectors.java:302)
        at com.genesyslab.mcr.smserver.channel.ChannelConnectors.configure(ChannelConnectors.java:185)
        at com.genesyslab.mcr.smserver.SmServer.initServer(SmServer.java:245)
        at com.genesyslab.mcr.smserver.SmServer.main(SmServer.java:130)
*** End of Trace ***
```

Password for your certificate is invalid

```
17:28:06.713 Std 43010 (channel-opensmpp).(oSmpp.initialize): error... failed to initialize ssl context: Get Key failed: Given final
block not properly padded
17:28:06.715 Std 42403 (PU channel-opensmpp).(<init>): exception caught and processed...
*** Start of Trace ***
java.lang.Exception: channel driver initiaization failed
        at com.genesyslab.mcr.smserver.channel.sms_mms.opensmpp.SmppDriver.initialize(SmppDriver.java:178)
        at com.genesyslab.mcr.smserver.channel.ChannelPU.<init>(ChannelPU.java:123)
        at com.genesyslab.mcr.smserver.channel.ChannelConnectors.processChannelSection(ChannelConnectors.java:302)
        at com.genesyslab.mcr.smserver.channel.ChannelConnectors.configure(ChannelConnectors.java:185)
        at com.genesyslab.mcr.smserver.SmServer.initServer(SmServer.java:245)
        at com.genesyslab.mcr.smserver.SmServer.main(SmServer.java:130)
*** End of Trace ***
```

Configuring privacy

Session mode

SMS sessions are implemented via chat sessions in Chat Server. Therefore, filtering and masking of PII (personally identifiable information) happens in Chat Server.

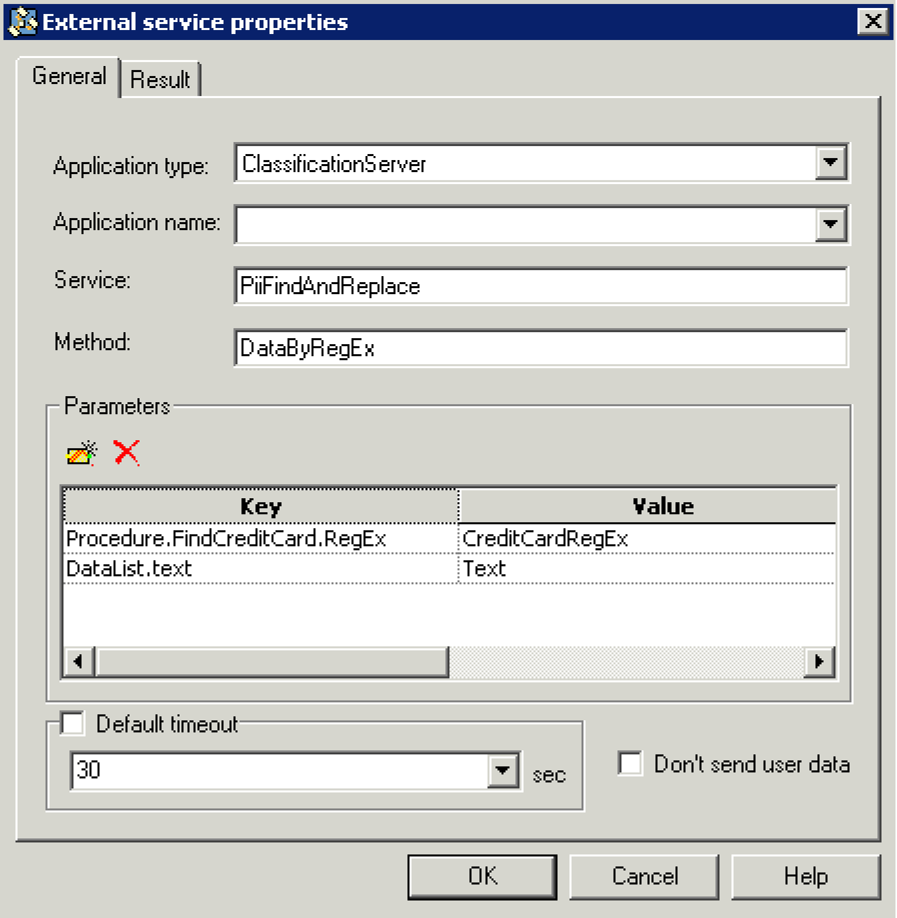
Chat Server reads PII rules from Universal Contact Server (UCS) and then applies these rules to messages within chat transcripts. Refer to the Masking Sensitive Data page in the Chat Server Administration Guide for more information. Also, refer to the Privacy Manager Guide for more information on using Privacy Manager to configure PII rules.

Page mode

For page mode, you must send PII requests from the business process. You can use the following ESP requests from the strategy:

- **DataByGroup**
- **DataByRule**
- **DataByRegEx**

The following is a sample:



External service properties

Application type: ClassificationServer

Application name:

Service: PiiFindAndReplace

Method: DataByRegEx

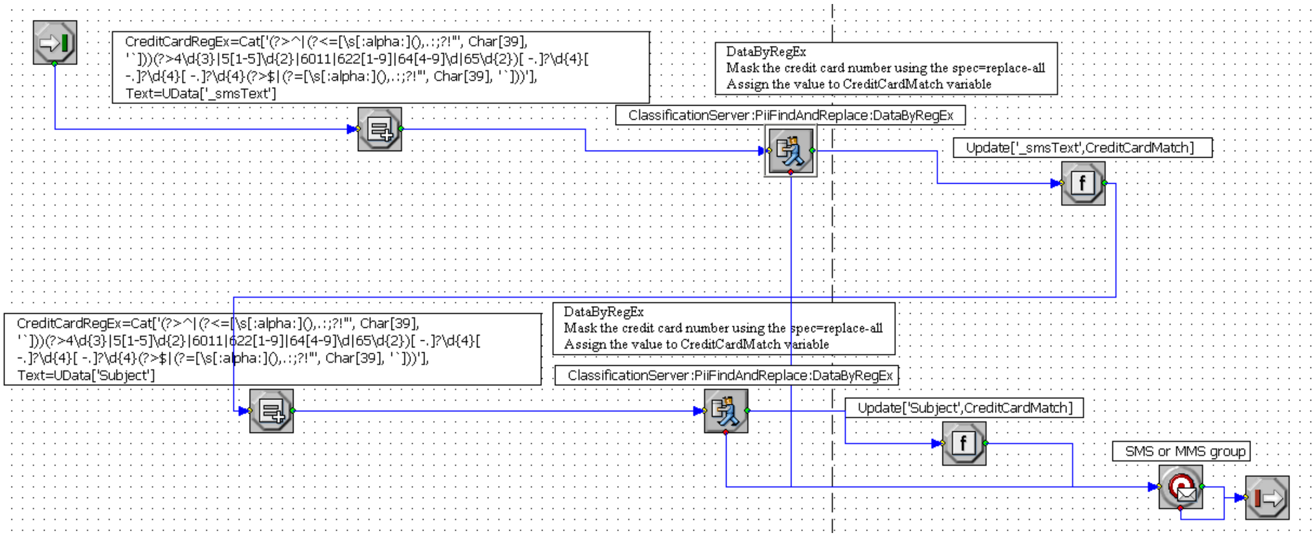
Parameters

Key	Value
Procedure.FindCreditCard.RegEx	CreditCardRegEx
DataList.text	Text

□ Default timeout

30 sec

□ Don't send user data



```
CreditCardRegEx=Cat['(?>^|(?<=[\s[:alpha:](),.:;?!"', Char[39],
'`]))(?>4\d{3}|5[1-5]\d{2}|6011|622[1-9]|64[4-9]\d|65\d{2})[ -.]?\d{4}[
-.]?\d{4}[ -.]?\d{4}(?>$|(?=[\s[:alpha:](),.:;?!"', Char[39], '`]))'],
Text=UData['_smsText']
```

DataByRegEx
Mask the credit card number using the spec=replace-all
Assign the value to CreditCardMatch variable

ClassificationServer:PiiFindAndReplace:DataByRegEx
Update['_smsText',CreditCardMatch]

```
CreditCardRegEx=Cat['(?>^|(?<=[\s[:alpha:](),.:;?!"', Char[39],
'`]))(?>4\d{3}|5[1-5]\d{2}|6011|622[1-9]|64[4-9]\d|65\d{2})[ -.]?\d{4}[
-.]?\d{4}[ -.]?\d{4}(?>$|(?=[\s[:alpha:](),.:;?!"', Char[39], '`]))'],
Text=UData['Subject']
```

DataByRegEx
Mask the credit card number using the spec=replace-all
Assign the value to CreditCardMatch variable

ClassificationServer:PiiFindAndReplace:DataByRegEx
Update['Subject',CreditCardMatch]
SMS or MMS group