Genesys Driver for SMS and MMS Guide

Messaging Applications 9.0.0

1/19/2025

Table of Contents

Genesys Driver for SMS and MMS

This document describes how to deploy and use the Genesys Driver for SMS and MMS.

What is in this document?

- Deploying the Driver—Describes how to deploy and configure the driver in your environment.
- Sample Business Processes—Provides sample business processes to use with the driver.
- Administering the Driver—Describes how to administer various functions and features of the driver.
- Security and Privacy—Provides information specific to security and privacy features of the driver.

How do agents use SMS and MMS?

Refer to the following topics in the Workspace Desktop Edition User's Guide:

- Handle An SMS or MMS Interaction
- Transfer An SMS or MMS Interaction

Deploying the Driver

Prerequisites

- Digital Messaging Server 9.0.002.06 or later

Note for customers upgrading from SMS Server

If you are upgrading from SMS Server, you can deploy Digital Messaging Server with the SMS/MMS driver on the same port and host as your previous SMS Server installation. After deployment is complete, you can start DMS with the SMS/MMS driver with its own Social Messaging Server Application type.

Deploy Digital Messaging Server

Follow the installation procedure for Digital Messaging Server.

Install the driver

1. Navigate to the folder that contains the installation package for Genesys Driver for SMS and MMS.
2. Run the installation for Genesys Driver for SMS and MMS:
   On Windows, run `setup.exe`

   On Linux, run `install.sh`
3. When prompted by the installation dialog, specify the proper Configuration Server parameters for your environment and select the desired Digital Messaging Server application.
4. After installation, locate the **SmsMmsDriver.cfg** configuration file in the **\<Digital Messaging Server application>\media-channel-drivers\channel-smsmms\Configuration** folder.
5. In GAX, open your Digital Messaging Server Application, go to the **Options** tab, and import options from the **SmsMmsDriver.cfg** file.

Configuration

All options are documented in the Options Reference. You must properly configure the x-jsms-config-file option before using the SMS/MMS driver. This option is required for MMS processing.

Configuring Chat Server

Genesys recommends that you use a separate Chat Server to handle SMS and MMS messages. Set the following Chat Server options:

- stop-abandoned-interaction = `never`
- transcript-auto-save = 1

Ensure that the following Chat Server options are set to their default values:

- use-contact-server = `true`
- session-restoration-mode = `none`

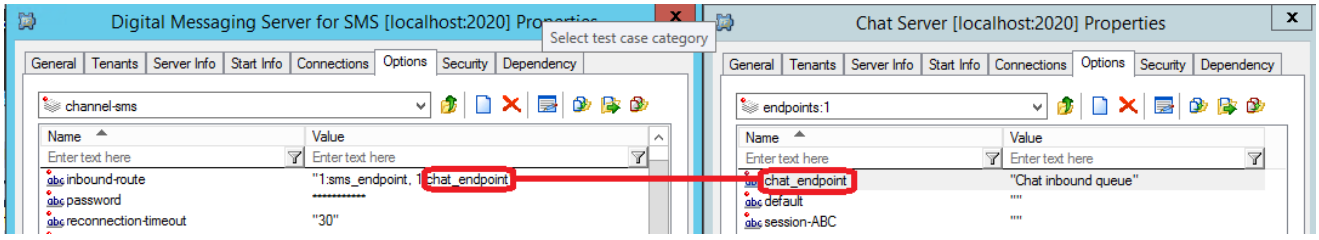
There is a limit on the concurrent chat sessions Chat Server can support at a time. It depends on the hardware and chat session scenario, but as a rule of thumb you can expect a maximum of 1,000 concurrent sessions on one Chat Server.

Endpoints

You must specify endpoints for inbound messages arriving from the media channel.

For example:

- **1:sms_endpoint**—Endpoint for regular interactions, which are processed in **paging** mode.
- **1:sms_endpoint**, **1:chat_endpoint**—Same as above, plus an additional endpoint that Chat Server uses to place interactions in **chat session** mode



Other configuration

- Genesys recommends that you increase the default value of the session-shutdown-timeout option in DMS to `1800` (30 minutes).
- Genesys recommends that you increase the default value of the xml-request-max-size option in Chat Server to `100` (kilobytes).

Uninstalling or upgrading the driver

To uninstall or upgrade the SMS/MMS driver, do the following:

1. Stop the DMS application with the Genesys Driver for SMS and MMS. This can be done for example via GAX (or **Services** if using Windows).
2. Uninstall the Genesys Driver for SMS and MMS application:
   - On Windows through **Control Panel > Programs and Features**.
   - On Linux by running the **./uninstall_custom.sh** script from the **media-channel-drivers/channel-smsmms/uninstall** folder.
3. Verify that the folder **channel-smsmms** was removed under folder **media-channel-drivers**.
4. Install the new version of Genesys Driver for SMS and MMS, using the instructions above on this page.
5. Start the DMS application with the Genesys Driver for SMS and MMS.

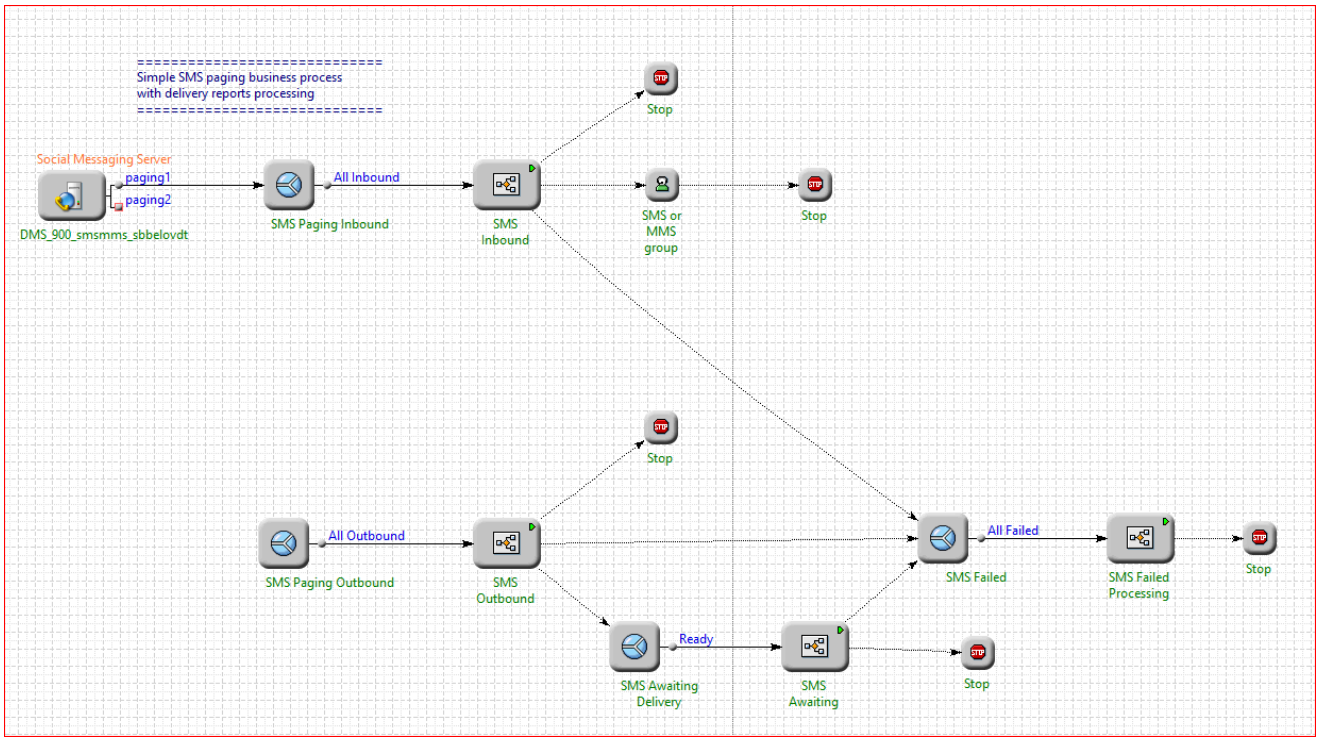
Sample Business Processes

> **Important**
>
> You might need to adjust business processes to work with your particular Genesys environment. For example, you might need to adjust the business processes to work with specific servers and agent groups that you have set up.

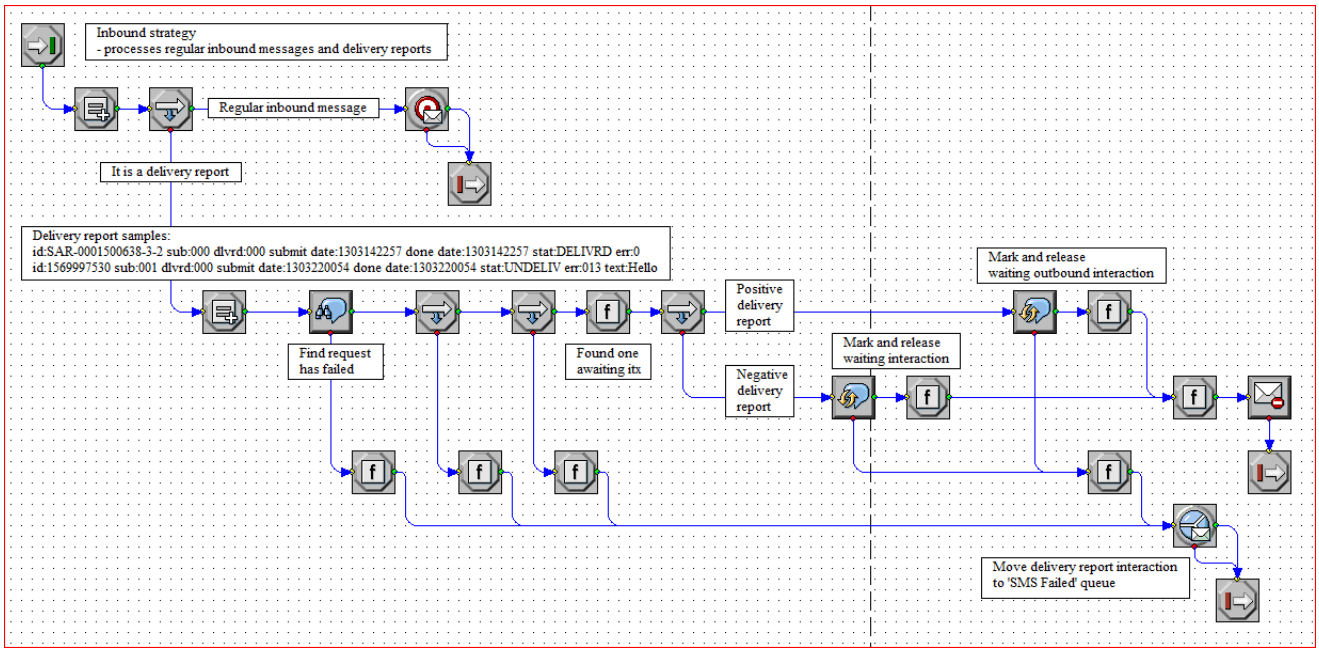
Genesys Driver for SMS and MMS includes the following sample business processes in the following folder: **<DMS_folder>/media-channel-drivers/channel-smsmms/Business Process Examples**.

Simple SMS Paging

The Simple SMS Paging business process shows how to receive and send SMS messages, and how to process SMS delivery reports. This business process uses a generic object to initiate the **SendSMS** request to an SMS driver/DMS pair. This generic object serves as an example on how to specify the ESP request.

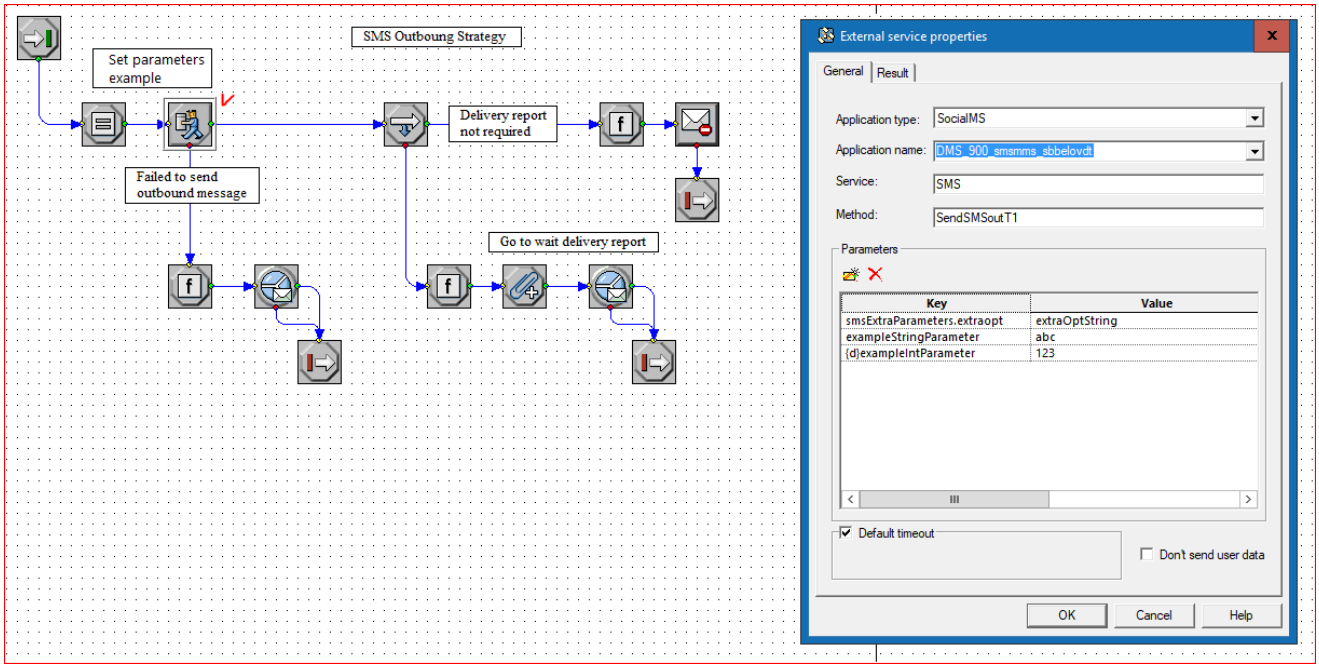


SMS Inbound

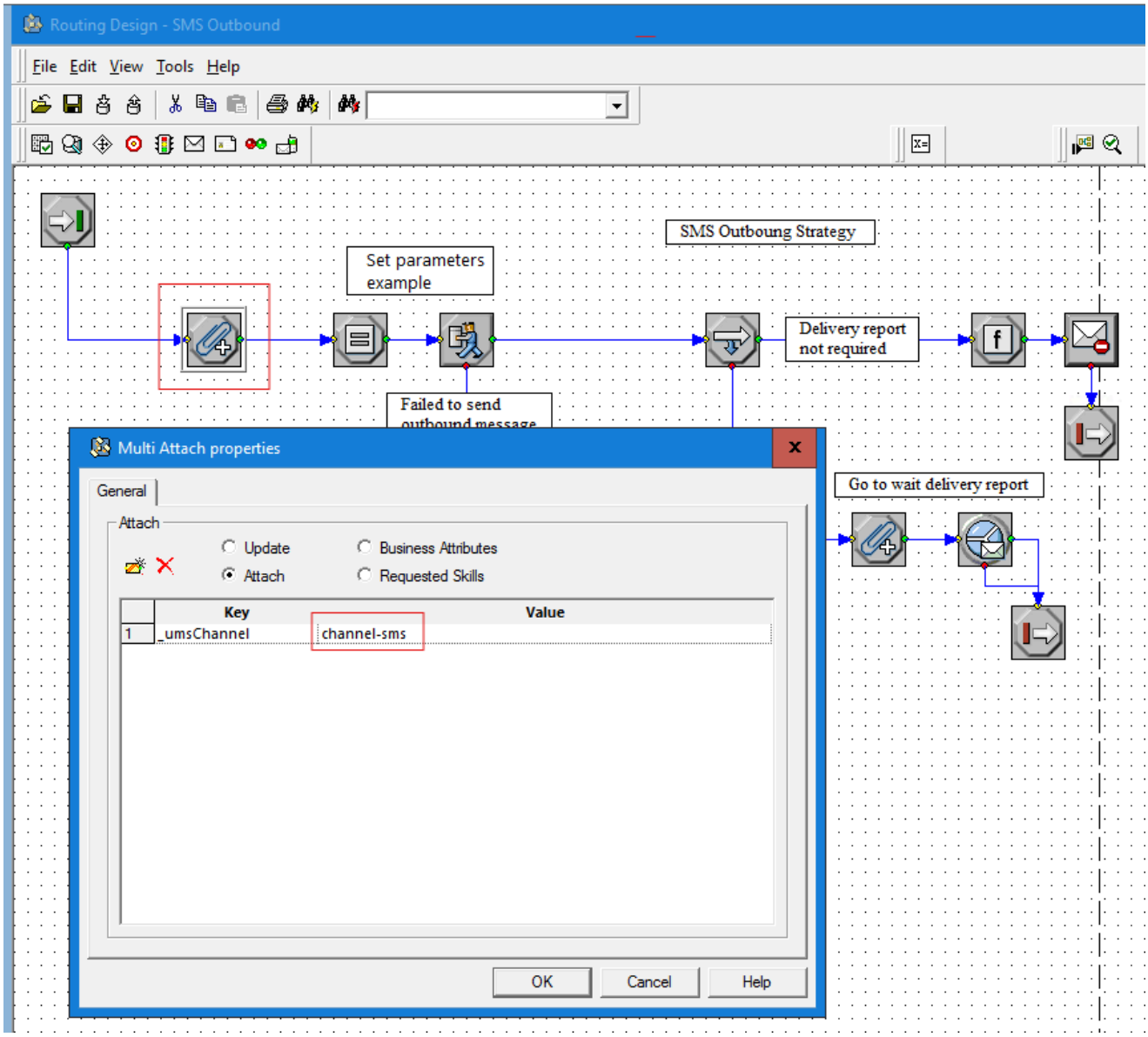


SMS Outbound

This business process uses the "External Service" IRD object to send outbound messages.

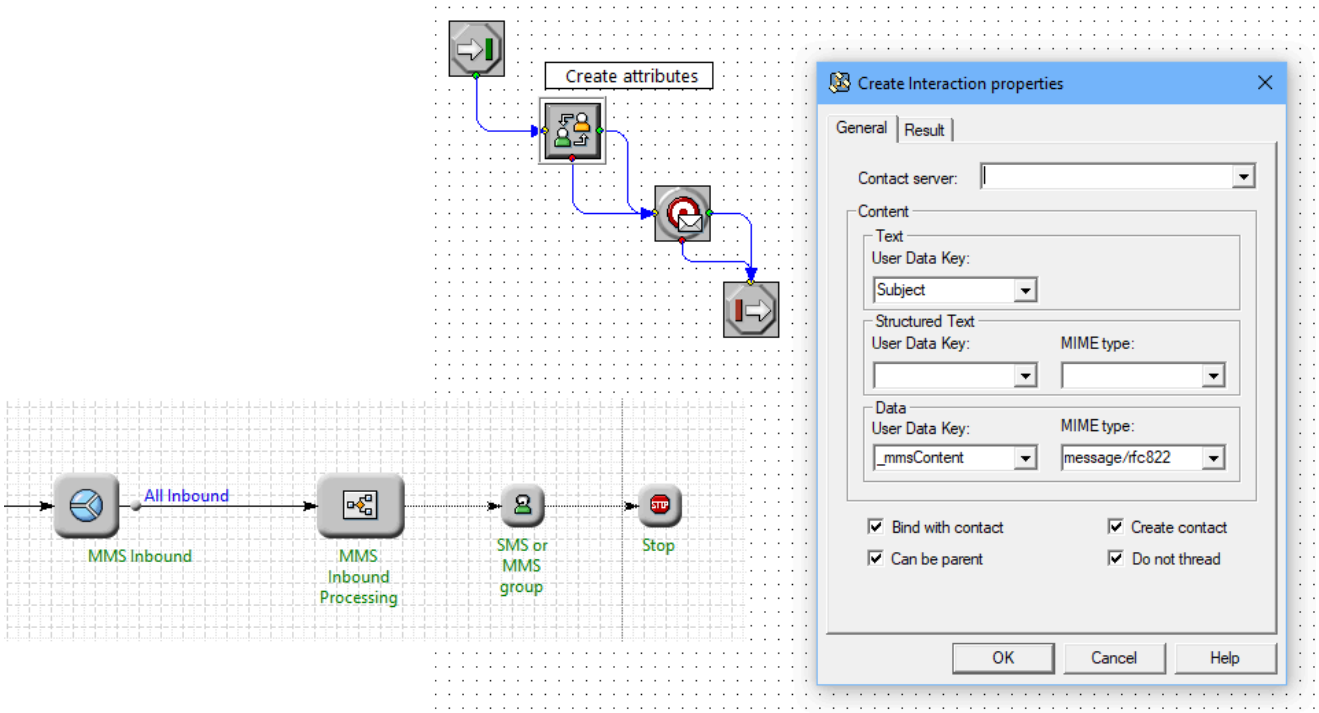


How to enable outbound paging messages: To enable outbound paging messages, please add **MultiAttach** block into strategy, as show on the screenshot below. Specify `_umsChannel = <sms channel name>`; use the channel name exactly as it appears in *Digital Messaging Server* settings.



Simple MMS

This business process shows how to receive MMS messages.


Create attributes
Create Interaction properties
General
Result
Contact server:
Content
Text
User Data Key:
Subject
Structured Text
User Data Key:
MIME type:
Data
User Data Key:
MIME type:
_mmsContent
message/rfc822
Bind with contact
Create contact
Can be parent
Do not thread
OK
Cancel
Help
All Inbound
MMS Inbound
MMS Inbound Processing
SMS or MMS group
Stop

Administering the Driver

This section provides information for administrators of Genesys Driver for SMS and MMS. In addition to the topics on this page, see also Supported SMPP v3.4 Operations.

How the driver handles empty messages

Sometimes, an SMS provider delivers empty messages (no text, no payload). These messages must be processed by an agent. To send empty messages to an agent for processing, enter text for the value of the x-smpp-empty-message configuration option. This text will be delivered to an agent as a content of the original empty message.

> **Important**
>
> The text you specify for the x-smpp-empty-message configuration option can optionally be an empty string.

How the driver masks sensitive data in logs

To mask sensitive data, perform the following two steps:

1. To activate SMS/MMS driver-specific filtering, you must:
   - Set the logging-filter-active configuration option to `true`.
   - Set the logging-filter-spec configuration option to `./media-channel-drivers/channel-smsmms/logging-filter-smsmms.json`.

   > **Important**
   >
   > Refer to the DMS Options Reference and Genesys Driver for SMS and MMS Options Reference for more information.

2. Apply the standard log filtering. Standard log filtering covers key-value pairs in User Data of ESP requests, which can contain sensitive information. Use `_smsText`, `Subject`, `FromAddr`, `PhoneNumber`, and `_smsDestNumber` key-value pairs for filtering. To filter out information in the target key-value pair, in the `log-filter-data` section, create an option with the same name as the key-value pair (for example, `Subject`) and set its value to `hide` or `skip`.

How the driver processes extra parameters for PDU

Some SMPP commands (or PDU – Protocol Data Unit) contain a set of optional parameters. Some SMS providers require these optional parameters to correctly process SMPP protocol. SMS Driver supports this functionality – you can define PDU's optional parameters on two levels:

- As a parameter of ESP request – Inside ESP request in a strategy, optional parameters are defined by the key-value pair `extraopt` in `Extra` optional parameters of `Send SMS Out` block.
- As a server option – In the server, optional parameters are defined by the value of the x-smpp-extraopt option in the `channel-any_name_for_sms` section.

If optional parameters are defined in the ESP request, they are used in PDU. If optional parameters are not defined in the ESP request, they are taken from the driver option. If optional parameters are not defined in either the ESP request or the server option, PDU is formed without optional parameters.

Whether defined in the ESP request or the server option, optional parameters are defined by the string of the format, as described below.

In SMPP, optional parameters are defined as a triplet (tag, length, value) called the TLV value. SMS Driver implements the following format of the string, representing optional parameters:

```
  tlvItems:
  [
    { tag:<tag value>, typ:<value type (i.e. byte or int or octets or short or strnz or strz)
>, val:<value> }
    . . . MORE TLV SPECIFICATIONS SEPARATED BY COMMAS
  ]
```

A parameter's type defines encoding and data size placed in PDU:

- `byte` is coded as 1-byte integer binary value
- `int` is coded as 4-bytes integer binary value
- `octets` is coded as a sequence of bytes, specified as a sequence of hex values
- `short` is coded as 2-bytes integer binary value
- `strnz` is coded as ASCII sequence with a length, defined by the string content without adding a terminating zero byte
- `strz` is coded as ASCII sequence ('CString' as in SMPP specification) with a length, defined by the string content with terminating zero byte added

Example:

```
{ tlvItems: [ {tag:4401, typ:octets, val:2a03b44c6d0f},  {tag:4402, typ:int, val:52173},  {tag:4403, typ:strz, val:abc1234},  {tag:4404,
typ:strnz, val:abc1234}, {tag:4405} ] }
```

Produces:

Tag (hex)	Length (hex)	Value (hex)
1131	0006	2a03b44c6d0f
1132	0004	0000cbcd
1133	0008	6162633132333400
1134	0007	61626331323334
1135	0000	<no value>

How the driver supports message-throttling by configurable rate

The SMS service provider might impose limits on the frequency with which they accept SMPP messages.

The SMS/MMS driver has configurable options that define parameters of communication with SMSC, including an option to control the maximum rate at which messages are sent to the SMS Center:

- x-smpp-sar-max-segments
- x-smpp-submit-max-rate
- x-smpp-submit-window-size
- x-smpp-response-max-waiting-time

Support of SMPP v3.4 Operations

Genesys Driver for SMS and MMS supports the following SMPP v3.4 operations:

Operation	Description
BIND_TRANSMITTER, BIND_TRANSMITTER_RESP, BIND_RECEIVER, BIND_RECEIVER_RESP, BIND_TRANSCEIVER, BIND_TRANSCEIVER_RESP	The purpose of the SMPP bind operation is to register an instance of an ESME (External Short Messaging Entity) with the SMSC (Short Message Service Center) system and request an SMPP session over this network connection for the submission or delivery of messages.
UNBIND, UNBIND_RESP	The purpose of the SMPP unbind operation is to deregister an instance of an ESME from the SMSC and inform the SMSC that the ESME no longer wishes to use this network connection for the submission or delivery of messages.
SUBMIT_SM, SUBMIT_SM_RESP	This operation is used by an ESME to submit a short message to the SMSC for onward transmission to a specified short message entity (SME).
DELIVER_SM, DELIVER_SM_RESP	DELIVER_SM is issued by the SMSC to send a message to an ESME. Using this command, the SMSC may route a short message to the ESME for delivery.
ENQUIRE_LINK, ENQUIRE_LINK_RESP	This message can be sent by either the ESME or SMSC and is used to provide a confidence check on the communication path between an ESME and an SMSC.

The protocol referred to in this section is described in Short Message Peer to Peer Protocol Specification v3.4, 12-Oct-1999 Issue 1.2.

SMS Inbound Messages Attributes

The following attributes are commonly used in regular workflows. However, your environment might have unique workflows that contain attributes that are not described on this page.

Name	Type	Comment	Example
MediaType	string	Media type of interaction	**smssession**
FromAddr	string	Source address (phone number)	**16501111111**
PhoneNumber	string	Source address (phone number - the same value as FromAddr)	**16501111111**
ToAddr	string	Destination address (phone number)	**16502222222**
Subject	string	Message subject (initial portion of message content)	**Test SMS message**
_smsSeqNumber	string	Sequence number from SMSC	**2654**
_smsInbDataEnc	string	Data encoding used in message	**1**
_smsSrcAddrTon	string	Source address TON (type of number)	**0**
_smsSrcAddrNpi	string	Source address NPI (numeric plan indicator)	**0**
_smsSrcNumber	string	Source address (phone number)	**16501111111**
_smsDestAddrTon	string	Destination address TON (type of number)	**0**
_smsDestAddrNpi	string	Destination address NPI (numeric plan indicator)	**0**
_smsDestNumber	string	Destination address (phone number)	**16502222222**
_smsText	string	Message content	**Test SMS message**
_umsChannel	string	Name of DMS channel that submitted message	**channel-sms-example**
_umsInboundIxnSubmittedBy	string	Name of application that submitted message	**DMS_900_smsmms**
_msg_CreatedAt	string	Creation date of message	**2019-01-28T02:39:56.035Z**

Security and Privacy

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

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.

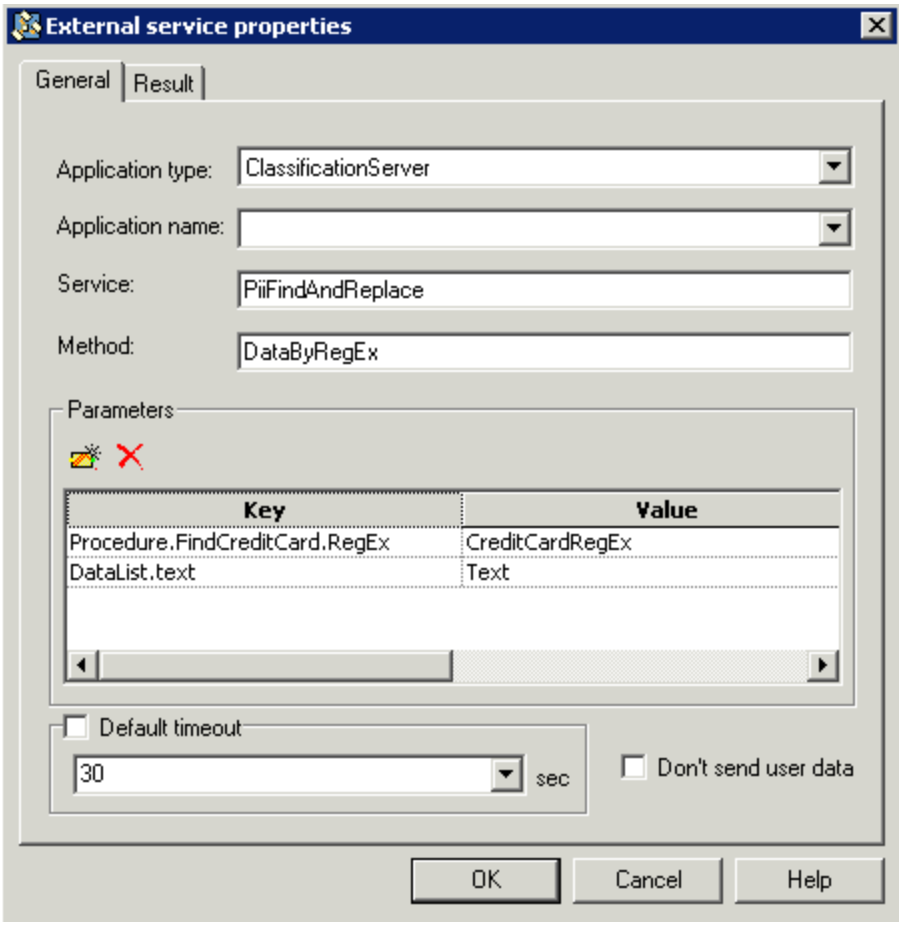
Chat Server reads PII rules from Universal Contact Server (UCS) and then applies these rules to messages within chat transcripts. Refer to the Masking Sensitive Data page in the Chat Server Administration Guide for more information. Also, refer to the Privacy Manager Guide for more information on using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

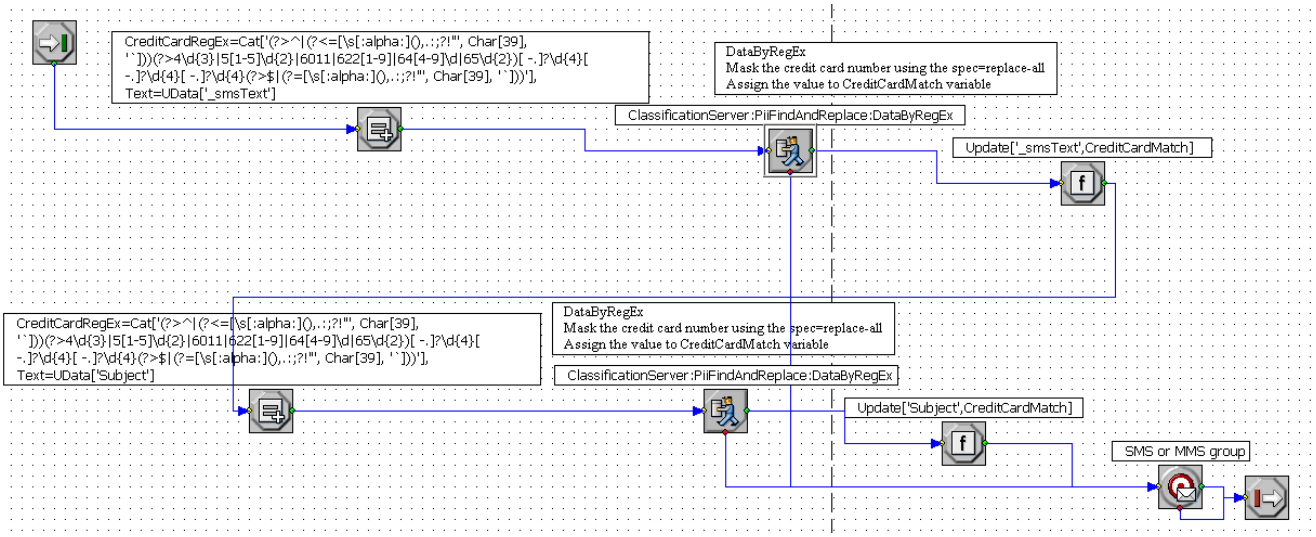
Page mode

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
General
Result
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
OK
Cancel
Help


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