Document Purpose
This document explains how SMS work in Campaign v6. The SMS connector allows to send and receive SMS with Campaign by connecting to a third party SMS provider with either the Netsize or the SMPP protocol.

The present document applies to the latest v6.1.1 builds only. While most use cases apply to older versions (like v6.02 or v5.x), a lot of bugs have been fixed in latest versions. It is highly recommended to upgrade to the latest build to avoid quirks and glitches. If you cannot do so, always READ THE CHANGELOG when you encounter problems before reporting them.

Since build 8670, the debug mode has been tremendously improved, providing much more details than in previous builds.

External resources

Wikipedia (English version) includes a number of valuable articles about the technical aspects of SMS:

- General article about SMS
- Article about SMPP
- GSM encoding character set

The official SMPP protocol specification is available here: SMPP Protocol Specification v3.4

A tutorial on analyzing SMPP captures with Wireshark is available here: SMPP protocol analysis using Wireshark (SMS)

Definitions

Here are a few definitions that you must know to read further:

- SMS-C: a gateway between internet and the mobile network. This is the server you connect to, hosted by the SMS provider.
- SMPP: the protocol used to communicate between Campaign and the SMS-C.
- MT: an SMS sent by Adobe Campaign to an end-user.
- SR: a delivery receipt sent automatically by the mobile when it receives an MT.
- MO: an SMS sent by an end-user to Adobe Campaign. For example, the end-user replies "STOP" to an MT to unsubscribe.
- SMPP RESP: a synchronous acknowledgement of an SMPP command. This is not the same as an SR, it only acknowledges transmissions between Campaign and the SMS-C.

SMPP protocol overview

The SMPP protocol is a network protocol defined by the SMPP Protocol Specification v3.4. It allows to exchange SMS between two servers.

SMPP opens 1 or 2 TCP connections, depending on its mode. All connections are always initiated by Campaign.

The SMPP protocol can work in 2 modes:

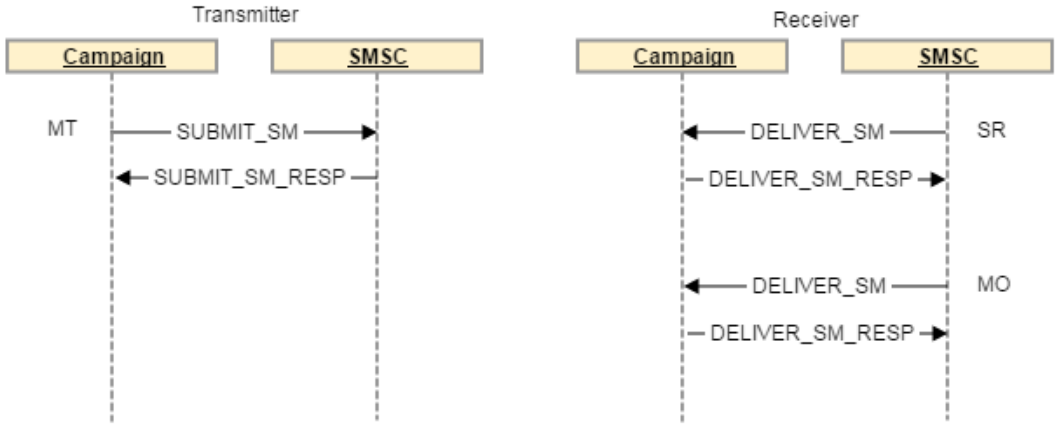
- Transmitter+receiver (often abbreviated TX+RX): two separate TCP connections are used for transmitting and receiving messages.
- Transceiver (abbreviated TRX): a single TCP connection is used for transmitting and receiving messages.

Note: Adobe Campaign v6.1.1 and earlier only support the TX+RX mode.

SMPP transmission units are called PDUs: a PDU contains a command, a status, a sequence number and data. Each PDU must be acknowledged by a SMPP RESP PDU (synchronous response). Requests may be pipelined: the sender can send many commands without waiting for RESP: the amount of requests that may be pipelined at any time is called the window.

In the separated transmitter+receiver mode, the connection used depends on the kind of message transmitted: the transmitter connection is used for MT, and the receiver connection is used for MO and SR. Warning: requests and responses for each kind of message are sent over the same TCP connection.

For example, when sending an MT, the transmitter connection is used and the RESP that acknowledges the MT is also sent through the transmitter channel. When you receive an MO (or an SR), the receiver connection is used to receive the MO and to send the RESP that acknowledges the MO.



SSL is not supported by most implementations, Campaign does not support SMPP over SSL. IP whitelisting and IPsec are the most used security measures.

External account

Mobile

The "Mobile" tab contains settings used by the web server to connect to the SMS-C. You must contact the SMS provider to check the information that you need to put in this form.

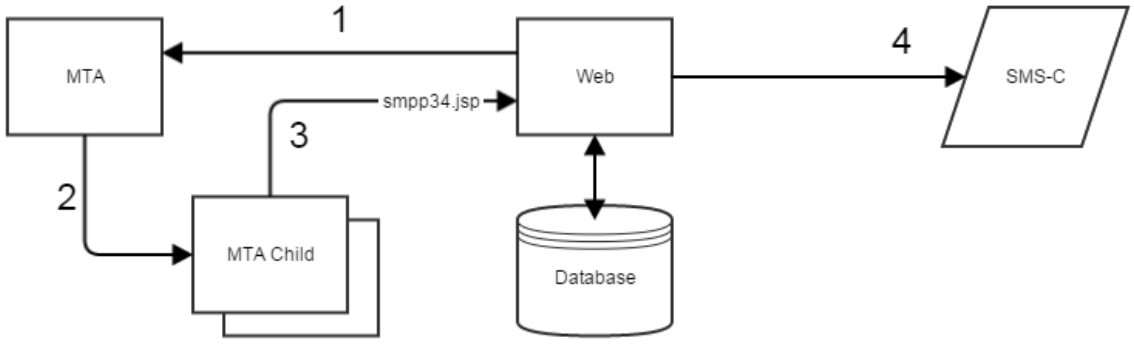
Connector

The connector URL depends on the provider:

- For Netsize, use http://HOST/nms/jsp/netsize.jsp
- For SMPP connectors (mBlox, Sybase365, Tele2, ...) use http://HOST/nms/jsp/smpp34.jsp
- For custom connectors, the URL may differ. Custom connectors are created, maintained and supported by consultants, not by R&D.

The HOST is the web server of the instance as seen by the MTA and SMS processes. Usually, this is localhost:8080.

Outgoing traffic

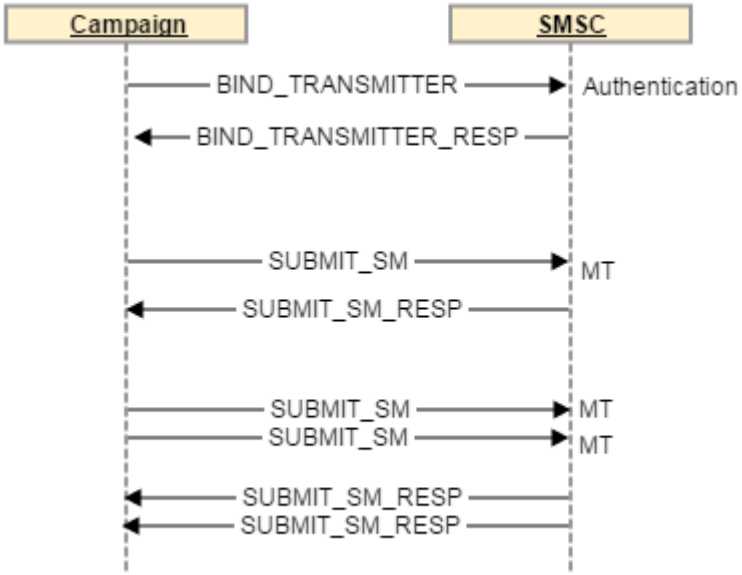


1. The marketing server sends the delivery parts to the MTA.

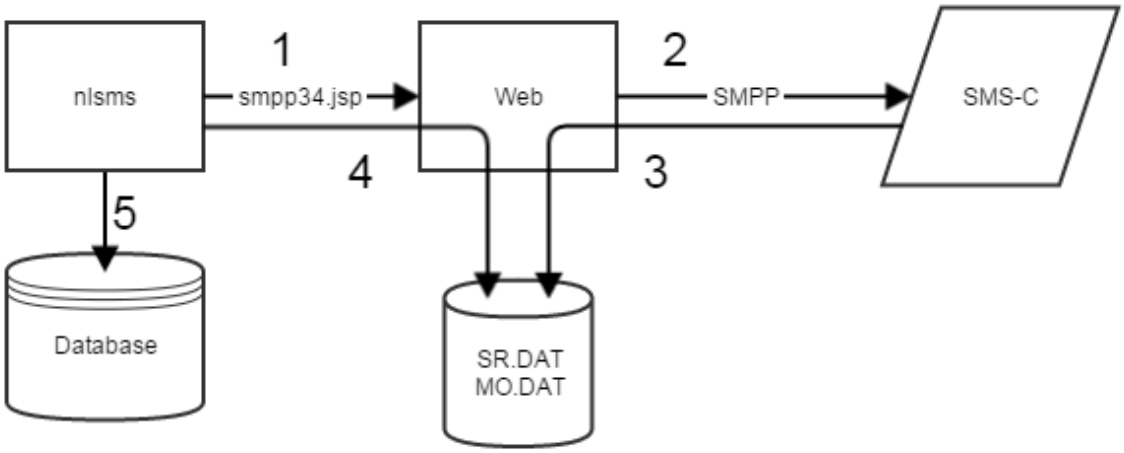
2. The MTA processes the delivery with the help of MTA child.
3. All messages are sent back to the web server to smpp34.jsp (or netsize.jsp). This step depends on the connector URL setting of the external account. Uses a SOAP call of the "deliveryPart" type.
4. The Java/JSP connector connects to the SMS-C and sends the SMS. This step depends on the mobile settings of the external account.

If MT are sent to the SMS-C, it means that external account settings are correct for the emitter.

Detailed sequence diagram for the SMPP protocol (step 4):

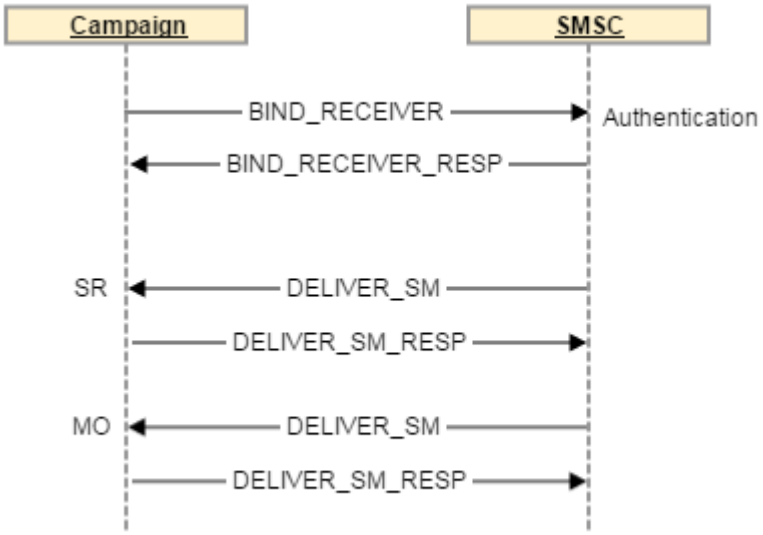


Incoming traffic



1. The nlsms process calls smpp34.jsp. This step depends on the connector URL setting of the external account.
2. The web server establishes an SMPP connection with the SMS-C. This step depends on the mobile settings of the external account.
3. The SMS-C sends SR and MO over the SMPP connection to the web server. These messages are stored into SR.DAT and MO.DAT on the web server. DAT file paths depend on host, port and login of the connector.
4. nlsms polls smpp34.jsp, which reads from the SR.DAT and MO.DAT files. This is done by SOAP calls with the "getStatus" method for SR and the "getMessages" method for MO.
5. nlsms updates the broadlog using the SR and inserts MO into inSms.

Detailed sequence diagram for the SMPP protocol (step 3):



DAT files contain incoming traffic, they contain raw data from the network socket, with only a few extra headers. To keep track of which part of the DAT file has already been processed and which part is still pending, an option named Mobile_<login>_<ip>_<port>_<machine>_SR (or _MO) is created containing the index of the next message to process.

These files offer a way to transfer data between the web server and the sms process. When the web process needs to store incoming traffic, it creates DAT files if they do not exist, or updates them if they already exist.

If DAT files contain new incoming traffic, it means that external account settings are correct for the receiver. To check this quickly, generate new traffic with a unique value (for example a phone that was never used before on this account) and search for the number in DAT files: if the phone number was added to the DAT file, you will know that the SMPP connection is working.

WARNING
If you delete, move or replace these files, you should restart the WEB and SMS processes because these processes keep the file open all the time, so that a restart is needed to refresh the file descriptor in process memory.

WARNING
If you delete, move or replace these files, remember to set Mobile_* options accordingly to point to the new index. If you have removed or reset the file, set the index to 0.

WARNING
Since the DAT file path is computed using host, port and login, each external account must have a unique combination of host, port and login. If this condition is not met, the files may be corrupt and traffic will not be routed to the right external account.

Troubleshooting

The aim of this troubleshooting section is to solve the most common problems with the connector, isolate problems properly and, if you hit a bug, give meaningful feedback to R&D.
Since the data path is very long and complex, make sure that you monitor every step in real-time when doing troubleshooting (using "tail -f" on log files, "tcpdump -X" on SMS-C ports, ...). If you troubleshoot with real-time feedback, it will be easier to spot which step fails.

The most important question you need to answer is "What made the problem occur", which is equivalent to "What would I need to do to create a new instance from scratch with the same symptoms". Finding the root cause is essential before opening a ticket: it speeds up the process of fixing problems tremendously, and people will reply much faster to the ticket.

Build number and changelogs

The SMS underwent many small improvements over time with each new build of v6.1.1. Upgrading to the latest build is therefore highly recommended. Especially, the "15.12" build (build number 8710 and higher) provides a much more useful (and verbose) debug output when the debug mode is enabled.
If you cannot upgrade to the latest build, always check the full changelog of the product to ensure that the problem you encounter has not already been fixed.

When reporting problems, always provide build numbers of all the servers (marketing as well as mid-sourcing, if applicable).

Isolating the faulty external account

Finding which SMS external account(s) has (have) problems will help you reduce the scope of the next steps. You will find problems faster because most troubleshooting operations need to be done for each suspicious external account.

Enable SMS external accounts one by one:

- Disable all SMS external accounts
- Enable the first account only
- Restart the web and sms processes to apply changes immediately
- Test whether the problem occurs
- Disable this account, enable the next one, restart web+sms once more and test again

... and so on for each SMS external account.

If no problem occurs when only one account is enabled at a time, you may have a conflict between different accounts. See the section "Identifying conflicts between accounts" below.

> In most cases, errors caused by a bug appear on all external accounts of the same type, and errors caused by a misconfiguration may not have a correlation between different accounts.

Identifying faulty steps when sending MT

- First, check if the delivery is running. If the delivery stays in pending status, this is not an SMS problem, but a generic delivery problem. Refer to the documentation about deliveries or use delivery troubleshooting procedures which are beyond the scope of this document.

- Check then if the MTA is running properly. Check the MTA log file, you should see your delivery in progress.

- Check if the MTA connects to the correct web server. This corresponds to step 3 in the MT diagram above. The MTA connects to the URL defined in the "Connector" section of the external account. To do this, enable the debug mode (see below) and check the web.log file for "deliveryPart" SOAP calls. Alternatively, you can analyze a network capture.
When checking debug traces, check that you see correct information in the SOAP calls (check that text is readable, recipient phone numbers are correct, ...).

- Check that the web server is successfully connected and bound to the SMS-C.
- The debug message telling us that the connector is connected successfully is "opened client tcp/ip connection to 10.1.2.3 on port 10000" (of course, IP and port will match your external account configuration).
- The typical debug message that indicates a successful bind looks like "Got response(?) pdu (bindresp: (pdu: 24 80000002 0 1) XXXXXXXX)": what is important here is to find "bindresp" and "80000002 0" on the same line.

- If the connection is closed immediately before or after receiving the bind response, it may indicate a problem with IP whitelisting on the SMS-C: contact the provider.

- Check that MTs are actually sent to the SMS-C. For this the most reliable way is to capture network traffic. If you are unable to do it, check that SOAP calls succeed in the web.log file (you need the debug mode for that). Check that MTs are acknowledged correctly in the log or the capture.

Identifying faulty steps when receiving SR or MO

- First, check that the sms process is up and running: check in the process list in the monitoring section of the console home page (you can also use the "ps aux" linux command). Restarting the sms process will make it poll the web server immediately, which helps when doing a lot of tests during troubleshooting.

- Then, check that the sms process sends SOAP requests to the web server. This corresponds to steps 1 and 4 in the MO/SR diagram above. The SMS process connects to the URL defined in the *Connector* section of the external account. For this, enable debug (see below) and check web.log. The SMS process sends "getStatus" SOAP calls to check for SR, and "getMessages" SOAP calls to check for MO.

- Check that the web server is successfully connected and bound to the SMS-C. If you have a version with build number 8670 or later, you can trace the bind operation in the log with the debug mode enabled:

  - The debug message telling us that the connector is connected successfully is "opened client tcp/ip connection to 10.1.2.3 on port 10000" (of course, IP and port will match your external account configuration).
  - The typical debug message that indicates a successful bind looks like "Got response(?) pdu (bindresp: (pdu: 24 80000001 0 1) XXXXXXXX)": what is important here is to find "bindresp" and "80000001 0" on the same line.

If you have a build older than 8670, you can either ask the SMS-C provider, or use a network capture and search for a bind_transmitter or bind_receiver PDU.

- If the connection is closed immediately before or after receiving the bind response, it may indicate a problem with IP whitelisting on the SMS-C: contact the provider.

- Check that the SMS-C is sending valid data. This is the tricky part! You probably want to do this check as a last resort because this is by far the most time-consuming task in the present list. See "Capturing and checking that network data is valid and well formatted" below for more details.

- Check that the DAT files are present in the directory nl6/var/<instance>/mobile/<IP><login><port>/.
To check that these files are well formatted, you can extract strings from them with the UNIX "strings" command (available on Debian and most distributions). You should find text and phone numbers in it. If the file is blank or if it is not updated when you should receive traffic, it means that you did not receive any data from the provider (double-check previous steps).
To be extra sure that these DAT files are correct, see the "Replaying DAT files on another platform" section below.

- In case of doubt about DAT files, back them up and delete them, then set the Mobile_<login>_<ip>_<port>_<machine>_SR and _MO options to 0. Then restart both SMS and WEB servers. This will reset DAT file parsing, which may help solving the problem. If it solves your problem, please attach the problematic backups to a ticket for further analysis by R&D, you may have found a bug! (See last section for reporting problems.)

- As explained in the "Identifying conflicts between accounts" section below, a failed DAT file may prevent other accounts from working.

- Check that the SMS process can insert MOs in the InSMS table.

Identifying conflicts between accounts

You may observe interactions between accounts, and these interactions may lead to conflicts or make one external account block others.

The most common conflict is 2 accounts having the same host, port and system_id information.

> **WARNING**
> Since the DAT file path is computed using host, port and login, each external account must have a unique combination of host, port and login. If this condition is not met, the files may be corrupt and traffic will not be routed to the right external account.

If a DAT file is corrupt, it may prevent other external accounts from working correctly. The SMS connector will stop all processing if a problem is encountered in a DAT file. Thus, all DAT files usually processed after the corrupt one will not be processed, despite being perfectly fine. However, they will be filled, so you will see new traffic in them but they will grow forever and not be processed until the corrupt DAT file is removed or fixed.

Remember to restart SMS+WEB processes and set Mobile_* index options when working with DAT files.

Connecting from another server

If you cannot qualify an error, the most radical way to do it is to create a new instance from scratch, with no customization, and connect it instead of the normal instance. If the problem persists, you may have found either a bug or an incompatibility with the SMS provider. Check that network data is correct (see the corresponding section below).

If the new instance works correctly, apply customizations one by one to match the source platform until it fails, that way you will know what you will have to revert on the initial platform. You may have to try many different configurations.

Common tasks

In the following part of the document, you will find instructions on how to perform some tasks useful to use or troubleshoot the SMS connector.

Analyzing log files

Since the SMS connector is hosted by the web server of Adobe Campaign, the most important log file is the web.log file (found in nl6/var/default/log/web.log). The debug mode needs to be enabled to have information in this file.

There are a few error messages that are normal, especially messages on transient errors telling that the web server is unavailable (smpp34.jsp or netsize.jsp URLs cannot be reached). This may be caused by the web server restarting or the web server being overloaded. If this happens too often, this is a platform problem, you need to optimize workload or improve performances in general.

The mta.log and sms.log provide extra information. The sms.log file is especially important when debugging insertions in the InSMS table. You may need to enable verbose output for the sms process or even sql trace / queryplans for better output.

Debug mode

To enable the debug mode for the SMS connector, add "?debug=true" to the connector URL.
For example, if the URL was "http://localhost:8080/nms/jsp/smpp34.jsp", it should become "http//localhost:8080/nms/jsp/smpp34.jsp?debug=true".

If you want to debug sending MT, you will have to restart the WEB and MTA processes. If you want to debug receiving MO and SR, you will need to restart WEB and SMS processes.

Build(s) 8670 and later trace the whole SMPP data stream in the log when put in debug mode, this is very verbose but avoids the need for a network capture in most common cases.

Resetting DAT files

The DAT files are *.dat files stored in the directory nl6/var/<instance>/mobile/<IP><login><port>/.

The MO DAT files contain incoming SMS that have not been processed. The SR DAT files contain SR that have not been processed. If you reset these files, you may lose messages that were buffered inside the files. See the section "Replaying DAT files on another platform" if you wish to replay a backup DAT file in order to know if its content is important or not.

In case of doubt about DAT file content (possible corruption), apply the following procedure (you can apply the procedure either for the MO.dat file, the SR.dat file, or both):

- Back up the file and delete it.
- Set the Mobile_<login>_<ip>_<port>_<machine>_SR (or _MO, depending on the file you deleted) option to 0. This is an option found in Administration/Options in the web interface.
- Restart both SMS and WEB servers. This will reset DAT file parsing, which may help solving the problem.

If this resetting solves your problem, please attach the problematic backups to a ticket for further analysis by R&D, you may have found a bug! (See last section for reporting problems.)

Replaying DAT files on another platform

Replaying DAT files may be needed if you backed up a DAT file with important data (like processing "STOP" MO that is enforced by the law in many countries) and you don't want to lose its content. It may also help to isolate or reproduce problems on another instance.

- Backup/remove currently existing DAT files on the replay instance.
- Connect your instance to an idle SMS-C (that step is necessary to create the DAT files). It is recommended to use an SMPP simulator (free SMPP simulators are available online).
- Start or restart the web+sms processes.
- Locate the newly created DAT files for MO and SR. Overwrite them with the files you need to replay.
- Reset the counter Mobile_<login>_<ip>_<port>_<machine>_SR (or _MO) option.
- Restart again the web+sms process.
- The MO.dat and SR.dat files should be read, with the data contained in them.

Capturing and checking that network data is valid and well formatted

To check network data, you must analyze a network capture.

The capture must be done on the machine that hosts the SMS connector. That is the target of the connector URL defined in the external account. Usually, this is the mid web server.

To capture data using tcpdump, find the port used to communicate with the SMS-C, then use the following command (example for port 12345):

```
tcpdump -i any -w outfile.pcap tcp port 12345
```

The file outfile.pcap can then be opened using Wireshark. The technote SMPP protocol analysis using Wireshark (SMS) gives more detail about analyzing SMPP traffic.

If you've found a problem in the network capture, file a ticket to R&D and attach the file with a short description of the problem.

> **Note**
> When sending pcap files to R&D for analysis, check that you really sent a full capture (using the -w command-line switch) and not the standard output of tcpdump (using the > shell construct). The file must be readable by Wireshark if you need to check it before attaching it to the ticket.

How to report a problem

Reporting a problem about SMS requires special attention because of the complexity of the connector and the fact that there is limited visibility from outside the product.

Before reporting a problem, check with your SMS provider. They may help you solve your problem.

Please make sure the ticket includes the following elements:
- The build number of the marketing and mid-sourcing (if applicable) servers.
- The web.log file with the debug mode enabled. Put the log file that corresponds to the time when the problem first occurred. You may have to reproduce the problem with the debug mode enabled to catch that information.
- If applicable, put the "DAT" files.
- If the problem is between the SMS-C and the connector, attach network captures done on the web server when the problem occurs. Never trust network captures from an SMS provider, there might be a proxy in between that alters traffic.