

## Adobe Campaign - Technote

# SMPP protocol analysis using Wireshark (SMS)



### Document Purpose

Help analyzing SMPP traffic using Wireshark. Give hints about common caveats and oddities of the SMPP protocol and its implementations.

Most high-throughput SMS-C are compatible with the SMPP protocol version 3.4. This protocol allows sending SMS and receiving information about the delivery of these SMS. The SMPP protocol is documented in the [SMPP Protocol Specification v3.4](#) available on the internet as a PDF document.

The present technote is not a substitute for this specification, but it will try to give practical tips on how to interpret the protocol specification and match it with the Wireshark display, in order to help troubleshooting problems between Adobe Campaign and the SMS-C partner.

Since the SMPP protocol contains many different parts left to the interpretation of the implementor, there are differences between different SMS-C.

When troubleshooting problems, **always contact the SMS-C partner** to obtain information or to help you double-check what you see. If the SMS-C replies with an error, only the SMS-C partner will be able to tell you why it replied with the error. If you are using an SMPP simulator instead of connecting to a real SMS-C, you should use the source code (and perhaps use a debugger) to understand precisely what's going on.

## Capturing network traffic without Wireshark

If you don't have direct access to the machine, it may be necessary to capture using command-line tools like tcpdump.

If you already know the TCP port of the connection, put the correct filter to avoid capturing all traffic.

Here is a sample tcpdump command-line to capture port 12345 to outfile.pcap:

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

The file outfile.pcap can then be opened in Wireshark for further analysis.

## Wireshark handling

This technote assumes that you're familiar with the basics of Wireshark (capturing packets, defining simple filters, reading packet details, ...). A brief introduction is available on [howtogeek - How to Use Wireshark to Capture, Filter and Inspect Packets](#).

To filter out SMPP traffic in Wireshark, there are 3 important features:

- Use a display filter on the port of the SMS-C. For example, if the SMS-C uses port 10000, use the following filter:

```
tcp.port == 10000
```

- To isolate packets by phone number or by text content, use the search feature with the following settings:

Find

By: ☐ Display filter ☐ Hex value ☒ String

Filter:

Search In:

☐ Packet list

☐ Packet details

☒ Packet bytes

String Options

☐ Case sensitive

Character set:

ASCII Unicode & Non-Unicode

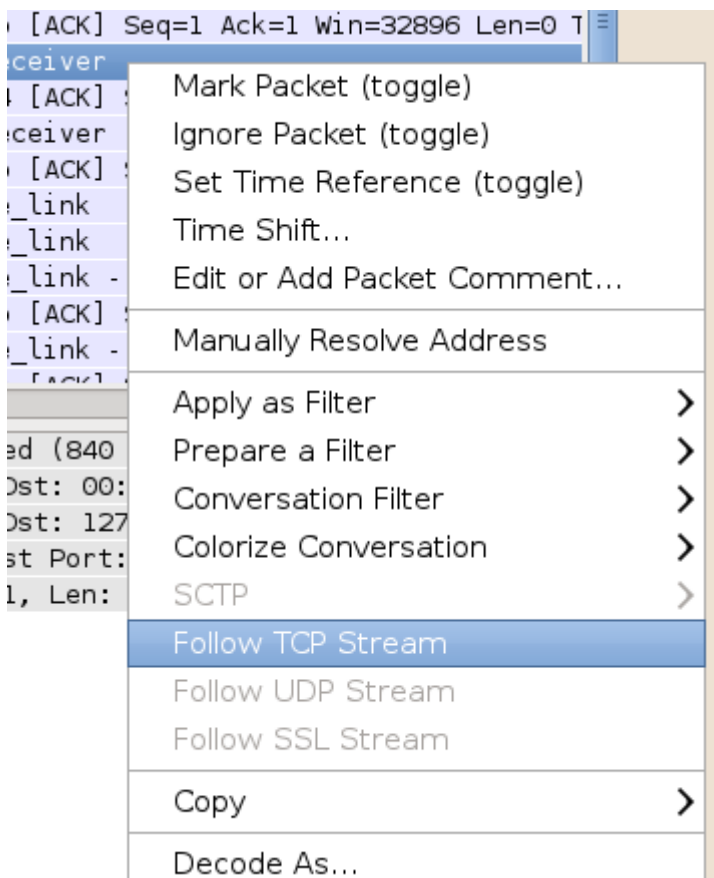
Direction:

☐ Up

☒ Down

Aide Annuler Rechercher

- Use the *Follow TCP stream* tool to isolate the stream you are working on. Close the red/blue text window that pops up because it is only useful for text protocols, which is not relevant to SMPP.



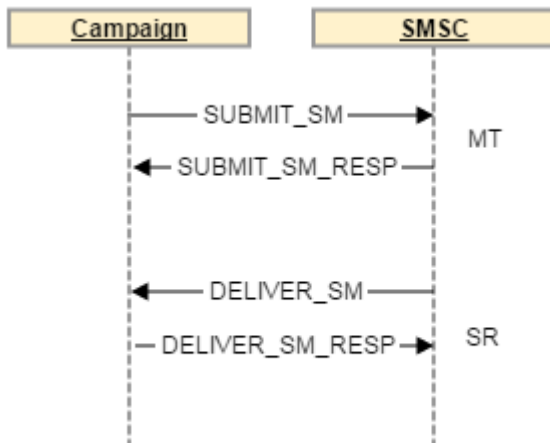
## The SMPP protocol

The protocol works over TCP and is fully binary, meaning that special tools like Wireshark (easy) or a hexadecimal editor (hard) are required to decipher the content of the stream. The stream is made up of independent PDUs: each PDU is a message containing a

command, a status, a sequence number and other information based on the command.

Due to the nature of TCP as a stream protocol, a TCP packet may contain more than one PDU and PDUs may span over 2 or more TCP packets. Wireshark will reassemble PDUs correctly, so it is mostly transparent for the Wireshark user.

Here is an example of PDUs passing through the network when sending an MT, then receiving an SR:



The list of standard commands can be found in section 5.1.2.1 of the SMPP specification (*SMPP Command set*).

## SMPP responses

The SMPP protocol requires all commands to be acknowledged by a response PDU: BIND\_TRANSMITTER is acknowledged by BIND\_TRANSMITTER\_RESP, SUBMIT\_SM is acknowledged by SUBMIT\_SM\_RESP, etc.

There is a timeout for responses, it is typically 10, 30 or 60 seconds. The response may contain a positive acknowledgement (*command\_status* = 0) or an error (see 5.1.3 *command\_status*, table 5-2 in the SMPP specification for the list of standard errors). Most of the time, these responses are quick enough and a response timeout is almost never a problem.

Care should be taken to distinguish between SMPP response errors and SR error codes, they are not the same thing and the same error code may mean different things in the response error or in the SR error field. When reporting an error code, be very precise about where you found it because the meaning of the value depends on where it comes from.

## SMPP connection initialization

The SMPP connection starts by connecting using TCP. Then a BIND operation is sent by campaign, acknowledged by a BIND RESP. These operations are described in section 4.1 of the SMPP specification (*BIND operation*).

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: `tcp.port == 10000` Expression... Clear

No.	Time	Source	Dest	Protocol	Length	Info
344	16:45:12.492	10000	53823	TCP	74	ndmp > 53823 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
345	16:45:12.492	53823	10000	TCP	66	53823 > ndmp [ACK] Seq=1 Ack=1 Win=32896 Len=0 TS=
354	16:45:12.494	53823	10000	SMPP	105	SMPP Bind_transmitter
355	16:45:12.494	10000	53823	TCP	66	ndmp > 53823 [ACK] Seq=1 Ack=40 Win=32768 Len=0 TS=

Frame 354: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface 0

Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)

Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)

Transmission Control Protocol, Src Port: 53823 (53823), Dst Port: ndmp (10000), Seq: 1, Ack: 1, Len: 39

Short Message Peer to Peer, Command: Bind\_transmitter, Seq: 1, Len: 39

Length: 39

Operation: Bind\_transmitter (0x00000002)

Sequence #: 1

System ID: smppclient

Password: azerty

Version (if): 3.4

Type of number: Unknown (0x00)

Numbering plan indicator: Unknown (0x00)

```

0020  00 01 d2 3f 27 10 1d 36 df f6 60 01 bb c6 80 18  ...?'..6 ..`....
0030  01 01 fe 4f 00 00 01 01 08 0a 00 67 38 63 00 67  ...0....g8c.g
0040  38 63 00 00 00 27 00 00 00 02 00 00 00 00 00 00  8c...'..
0050  00 01 73 6d 70 70 63 6c 69 65 6e 74 00 61 7a 65  ..smppcl ient.aze
0060  72 74 79 00 00 34 00 00 00                                rty..4...

```

Identifies a system. (smpp.sys... Packets: 1292 Display... Profile: Default

The bind operation does the login/password check and exchanges information about the platform name, version and other fields described in the specification.

The login can be found in the `system_id` field.

In Campaign, you should see a BIND\_TRANSMITTER packet when initiating an MT transfer, and a BIND\_RECEIVER packet when `nlsms` triggers an MO/SR connection.

**Transmitter, receiver and transceiver:** The SMPP connector for Campaign v6 works in a separate transmitter/receiver mode: there are two TCP connections, one for transmitting MT and another for receiving MO and SR. Note that the TCP connection is always initiated by Campaign, even for the receiver mode.

SMPP also provides a transceiver mode, but this mode is not implemented in the SMPP connector for Campaign v6.

The SMPP connector uses multiple connections in parallel to transmit MT. This cannot be controlled because of the way the connector is designed.

## Receiving MO

When the receiver is bound, the SMS-C may send MO at any time. The MO is sent using a DELIVER\_SM PDU with bits 2-5 of *esm\_class* clear (often, *esm\_class* will be simply 0).

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: `tcp.port == 10000` Expression... Clear Apply Enregistrer

No.	Time	Source	Dest	Protocol	Length	Info
73	16:55:42.492	53824	10000	TCP	66	53824 > ndmp [ACK] Seq=33 Ack=33 Win=257 Len=0 TSval=6922143
105	16:55:50.618	10000	53824	SMPP	142	SMPP Deliver_sm
106	16:55:50.618	53824	10000	TCP	66	53824 > ndmp [ACK] Seq=33 Ack=109 Win=257 Len=0 TSval=692417
111	16:55:50.621	53823	10000	SMPP	83	SMPP Deliver_sm - resp: "Ok"

Frame 105: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface 0

Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)

Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)

Transmission Control Protocol, Src Port: ndmp (10000), Dst Port: 53824 (53824), Seq: 33, Ack: 33, Len: 76

Short Message Peer to Peer, Command: Deliver\_sm, Seq: 1, Len: 76

Length: 76

Operation: Deliver\_sm (0x00000005)

Sequence #: 1

Service type: (Default)

Type of number (originator): International (0x01)

Numbering plan indicator (originator): ISDN (E163/E164) (0x01)

Originator address: 33612345678

Type of number (recipient): International (0x01)

Numbering plan indicator (recipient): ISDN (E163/E164) (0x01)

Recipient address: 102245

.... ..00 = Messaging mode: Default SMSC mode (0x00)

..00 00.. = Message type: Default message type (0x00)

00.. .... = GSM features: No specific features selected (0x00)

Protocol id.: 0x00

Priority level: GSM: None ANSI-136: Bulk IS-95: Normal (0x00)

Scheduled delivery time: Immediate delivery

Validity period: SMSC default validity period

.... ..00 = Delivery receipt: No SMSC delivery receipt requested (0x00)

.... 00.. = Message type: No recipient SME acknowledgement requested (0x00)

...0 .... = Intermediate notif: No intermediate notification requested (0x00)

.... ...0 = Replace: Don't replace (0x00)

Data coding: 0x00

Predefined message: 0

Message length: 26

Message

```

0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 .....E.
0010 00 80 96 e3 40 00 40 06 a5 92 7f 00 00 01 7f 00 ....@.@. ....
0020 00 01 27 10 d2 40 8a 83 b0 10 3a 3e 55 4f 80 18 ..'.@.. :>U0..
0030 01 00 fe 74 00 00 01 01 08 0a 00 69 a7 8e 00 69 ...t.... .i...i
0040 9f 9f 00 00 00 4c 00 00 00 05 00 00 00 00 00 00 .....L. ....
0050 00 01 00 01 01 33 33 36 31 32 33 34 35 36 37 38 .....336 12345678
0060 00 01 01 31 30 32 32 34 35 00 00 00 00 00 00 00 ...10224 5.....
0070 00 00 00 1a 48 65 6c 6c 6f 20 77 6f 72 6c 64 2e ....Hell o world.
0080 20 54 68 69 73 20 69 73 20 61 20 4d 4f 2e This is a MO.

```

The actual message or data. (...) Packets: 175 Displayed: 23 Mark... Profile: Default

The DELIVER\_SM PDU must be replied to quickly by a DELIVER\_SM\_RESP PDU with the same *sequence\_number*.


## Sending MT

---

To send an MT, the transmitter must be successfully bound. Before anything else, check that the bind process has been carried out successfully.

The MT is sent in a SUBMIT\_SM PDU. The SMS-C should quickly reply with a SUBMIT\_SM\_RESP PDU: this response packet is special because it contains the ID of the message in the database of the SMS-C (always include this ID when talking to the SMS-C partner to help him find the message more quickly). This ID will be present in the SR and is the only way to match the MT with its corresponding SR.

The field *registered\_delivery* (described in section 5.2.17 of the specification) indicates to the SMS-C whether an SR is requested for this particular MT. If you do not receive SR for a specific message, check that the field is correctly set in the SUBMIT\_SM PDU.

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help									
									
Filter:		tcp.stream eq 6				Expression...		Clear	Apply
No.	Time	Source	Dest	Protocol	Length	Info			
2923	17:56:03.994	53839	10000	SMPP	176	SMPP Submit_sm			
2928	17:56:03.995	10000	53839	SMPP	84	SMPP Submit_sm - resp: "Ok"			
2929	17:56:03.995	53839	10000	TCP	66	53839 > ndmp [ACK] Seq=111 Ack=19 Win=257 Len=0 TSva			
2932	17:56:03.997	53839	10000	SMPP	176	SMPP Submit_sm			

```
2977 17:56:04.010 10000 53839 SMPP      84 SMPP Submit_sm - resp: "Ok"
2978 17:56:04.010 53839 10000 SMPP      396 SMPP Submit_sm, Submit_sm, Submit_sm
2981 17:56:04.020 10000 53839 SMPP      84 SMPP Submit_sm - resp: "Ok"
3004 17:56:04.060 53839 10000 TCP        66 53839 > ndmp [ACK] Seq=551 Ack=55 Win=257 Len=0 TSva
3005 17:56:04.060 10000 53839 SMPP      102 SMPP Submit_sm - resp: "Ok", Submit_sm - resp: "Ok"
3006 17:56:04.060 53839 10000 TCP        66 53839 > ndmp [ACK] Seq=551 Ack=91 Win=257 Len=0 TSva
3029 17:56:04.202 53839 10000 SMPP      83 SMPP Deliver_sm - resp: "Ok"
3044 17:56:04.239 10000 53839 TCP        66 ndmp > 53839 [ACK] Seq=91 Ack=568 Win=256 Len=0 TSva
3045 17:56:04.239 53839 10000 SMPP      83 SMPP Deliver_sm - resp: "Ok"
3046 17:56:04.239 10000 53839 TCP        66 ndmp > 53839 [ACK] Seq=91 Ack=585 Win=256 Len=0 TSva
3176 17:56:06.007 53839 10000 SMPP      83 SMPP Deliver_sm - resp: "Ok"

<----->
> Frame 2978: 396 bytes on wire (3168 bits), 396 bytes captured (3168 bits) on interface 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
> Transmission Control Protocol, Src Port: 53839 (53839), Dst Port: ndmp (10000), Seq: 221, Ack: 37
▼ Short Message Peer to Peer, Command: Submit_sm, Seq: 16, Len: 110
    Length: 110
    Operation: Submit_sm (0x00000004)
    Sequence #: 16
    Service type: (Default)
    Type of number (originator): Unknown (0x00)
    Numbering plan indicator (originator): Unknown (0x00)
    Originator address: source12345
    Type of number (recipient): Unknown (0x00)
    Numbering plan indicator (recipient): Unknown (0x00)
    Recipient address: 0610000001
    ....00 = Messaging mode: Default SMSC mode (0x00)
    ..00 00.. = Message type: Default message type (0x00)
    00.. .... = GSM features: No specific features selected (0x00)
    Protocol id.: 0x00
    Priority level: GSM: None          ANSI-136: Bulk          IS-95: Normal (0x00)
    Scheduled delivery time: Immediate delivery
    Validity period: Jun 28, 2015 15:56:03.000000000 UTC
    ....01 = Delivery receipt: Delivery receipt requested (for success or failure) (0x01)
    ....00.. = Message type: No recipient SME acknowledgement requested (0x00)
    ...0 .... = Intermediate notif: No intermediate notification requested (0x00)
    .......0 = Replace: Don't replace (0x00)
> Data coding: 0x00
    Predefined message: 0
    Message length: 35
    Message
▼ Optional parameters
    ▼ Optional parameter: dest_addr_subunit (0x0005)
        Tag: 0x0005
        Length: 1

<----->
0040 70 42 00 00 00 6e 00 00 00 04 00 00 00 00 00 00 00 pB...n.. .....
0050 00 10 00 00 00 73 6f 75 72 63 65 31 32 33 34 35   ....sou rcel2345
0060 00 00 00 00 30 36 31 30 30 30 30 30 31 00 00 00   ...06100 00001...
0070 00 00 31 35 30 36 32 38 31 35 35 36 30 33 30 30   ..150628 15560300
0080 30 2b 00 01 00 00 00 23 48 65 6c 6c 6f 20 53 4d o+.....# Hello SM
0090 53 20 77 6f 72 6c 64 20 66 72 6f 6d 20 41 64 6f S world from Ado
00a0 62 65 20 43 61 6d 70 61 69 67 6e 00 05 00 01 02 be Campa ign.....
00b0 00 00 00 6e 00 00 00 04 00 00 00 00 00 00 00 11   ...n....
00c0 00 00 00 73 6f 75 72 63 65 31 32 33 34 35 00 00   ...sourc el2345..
00d0 00 30 36 31 30 30 30 30 30 30 34 00 00 00 00 00 .0610000 004.....
00e0 21 25 20 26 22 21 25 25 26 20 22 20 20 20 2b     15062815 56020004
```

## Encoding of MT

Warning: encoding of SMS is a vast, complex subject with many traps and non-conforming implementations!

The first rule is **always contact the SMS-C partner in case of encoding problems**. Only they have precise knowledge of the encoding they support and special rules that may apply due to limitations in their technical platform. Make them check what you send to them and what they send back to you, it is the only path to a successful and stable interconnection.

SMS messages use a special 7 bits encoding, often called the GSM7 encoding. Wikipedia has [a good article about it \(GSM 03.38 in English\)](#).



In the SMPP protocol, GSM7 text will be expanded to 8 bits per character for easier troubleshooting. The SMS-C will pack it into 7 bits per character before it is sent to the mobile. This means that the *short\_message* field of the SMS may be up to 160 bytes long in the SMPP frame whereas it is limited to 140 bytes when sent on the mobile network (the most significant bit is simply discarded).

In case of encoding problems, here are some important things to check:

- First, make sure that you know what characters belong to which encoding. GSM7 is infamous for its partial support of diacritical marks (accents). Especially in French, where *é* and *ê* are part of GSM7, but *ê*, *â* or *î* are not. The situation is no better when it comes to Spanish.
- The C with cedilla (ç) is present only in upper case in the GSM7 alphabet, but some phones render it in lower case or "smart" case: the general recommendation is to completely avoid it and remove the cedilla (it is still very readable in French) or switch to UCS-2.
- **Do not use ASCII in SMS!** unless explicitly requested by the SMS-C partner: This encoding wastes space because it has 8-bit characters and less coverage than GSM7.
- Latin-1 is not always supported. Check the compatibility with your SMS-C partner before attempting to use Latin-1.
- National language shift tables are not supported by the Adobe Campaign v6 connector. You must use UCS-2 instead.
- UCS-2 and UTF-16 are often mixed by phones. This is a problem for people sending emoji and other rarely used characters not present in UCS-2.
- The GSM7 encoding is not supported by Wireshark: special characters will be displayed incorrectly. If you need to check whether a GSM7 string is properly encoded, you must compare hexadecimal codes with the GSM7 table.

The *data\_coding* field tells you which encoding is used. The only problem is that the value 0 means *default SMS-C encoding* in the specification, but in general it means GSM7. Check with the SMS-C partner what encoding is associated to *data\_coding* = 0 (Adobe Campaign only supports GSM7 for *data\_coding* = 0).

The maximum size of a message depends on its encoding. This table sums up all the relevant information:

Encoding	data_coding	Message size (characters)	Part size for multipart SMS	Available characters
GSM7	0	160	152	<a href="#">GSM7 basic character set + extension</a> (extended characters take 2 characters)
Latin-1	3	140	134	<a href="#">ISO-8859-1</a>
UCS-2 UTF-16	8	70	67	Unicode (varies from phone to phone)

## UDH

UDH (User Data Header) are small binary headers added to the text of an SMS. They can trigger special features like SMS concatenation, national language shift tables, logos/images (rarely used) or WAP push.

Since the UDH is part of the text field (*short\_message* SMPP field), it shortens the effective size of an SMS. For example, a concatenated SMS UDH will consume 6 bytes per SMS part (that's 6 real 8-bit bytes, **not** 7-bit characters), leaving enough room for only 152 7-bit characters per message part.

Again, English Wikipedia has nice articles about [User Data Header](#) and [Concatenated SMS](#).

To know whether a *short\_message* contains a UDH, check the bits 6 and 7 of *esm\_class* (see section 5.2.12 of the specification). Wireshark parses UDH in the interface and gives accurate information.

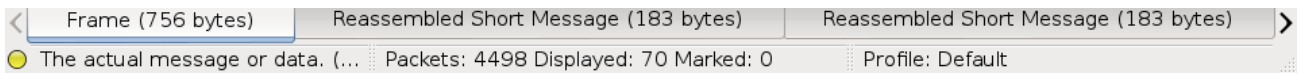
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: tcp.port == 10000 Expression... Clear Apply Enregistrer

No.	Time	Source	Dest	Protocol	Length	Info
67	09:46:14.486	54835	10000	TCP	66	54835 > namp [ACK] Seq=40 Ack=25 win=32896 Len=0 TSval=266889 TSecr=2
3863	09:46:26.536	54834	10000	SMPP	177	SMPP Submit_sm (Short Message Reassembled)
3872	09:46:26.541	10000	54834	SMPP	84	SMPP Submit_sm - resp: "Ok"
3873	09:46:26.541	54834	10000	SMPP	756	SMPP Submit_sm (Short Message Reassembled), Submit_sm (Short Message
3890	09:46:26.543	10000	54834	SMPP	84	SMPP Submit_sm - resp: "Ok"
3891	09:46:26.543	54834	10000	SMPP	756	SMPP Submit sm (Short Message Reassembled). Submit sm (Short Message

Recipient address: 0610000001  
 .... 00 = Messaging mode: Default SMSC mode (0x00)  
 ..00 00.. = Message type: Default message type (0x00)  
 01.. .... = GSM features: UDHI indicator (0x01)  
 Protocol id.: 0x00  
 Priority level: GSM: None ANSI-136: Bulk IS-95: Normal (0x00)  
 Scheduled delivery time: Immediate delivery  
 Validity period: Jun 29, 2015 07:46:25.000000000 UTC  
 .... 01 = Delivery receipt: Delivery receipt requested (for success or failure) (0x01)  
 .... 00.. = Message type: No recipient SME acknowledgement requested (0x00)  
 ...0 .... = Intermediate notif: No intermediate notification requested (0x00)  
 .... 00.. = Replace: Don't replace (0x00)  
 ▶ Data coding: 0x00  
 Predefined message: 0  
 Message length: 159  
 Message  
 ▶ Optional parameters  
 ▼ GSM Short Message Service User Data  
 ▼ UDHI Length: 5  
 ▼ IE Id: SMS - Concatenated short messages, 8-bit reference number (0x00): message 103, part 1 of 2  
 Message identifier: 103  
 Message parts: 2  
 Message part number: 1  
 ▼ [2 Short Message fragments (183 bytes): #3873(153), #3863(30)]  
 [Frame: 3873, payload: 0-152 (153 bytes)]  
 [Frame: 3863, payload: 153-182 (30 bytes)]  
 [Short Message fragment count: 2]  
 [Reassembled Short Message length: 183]  
 Short Message body

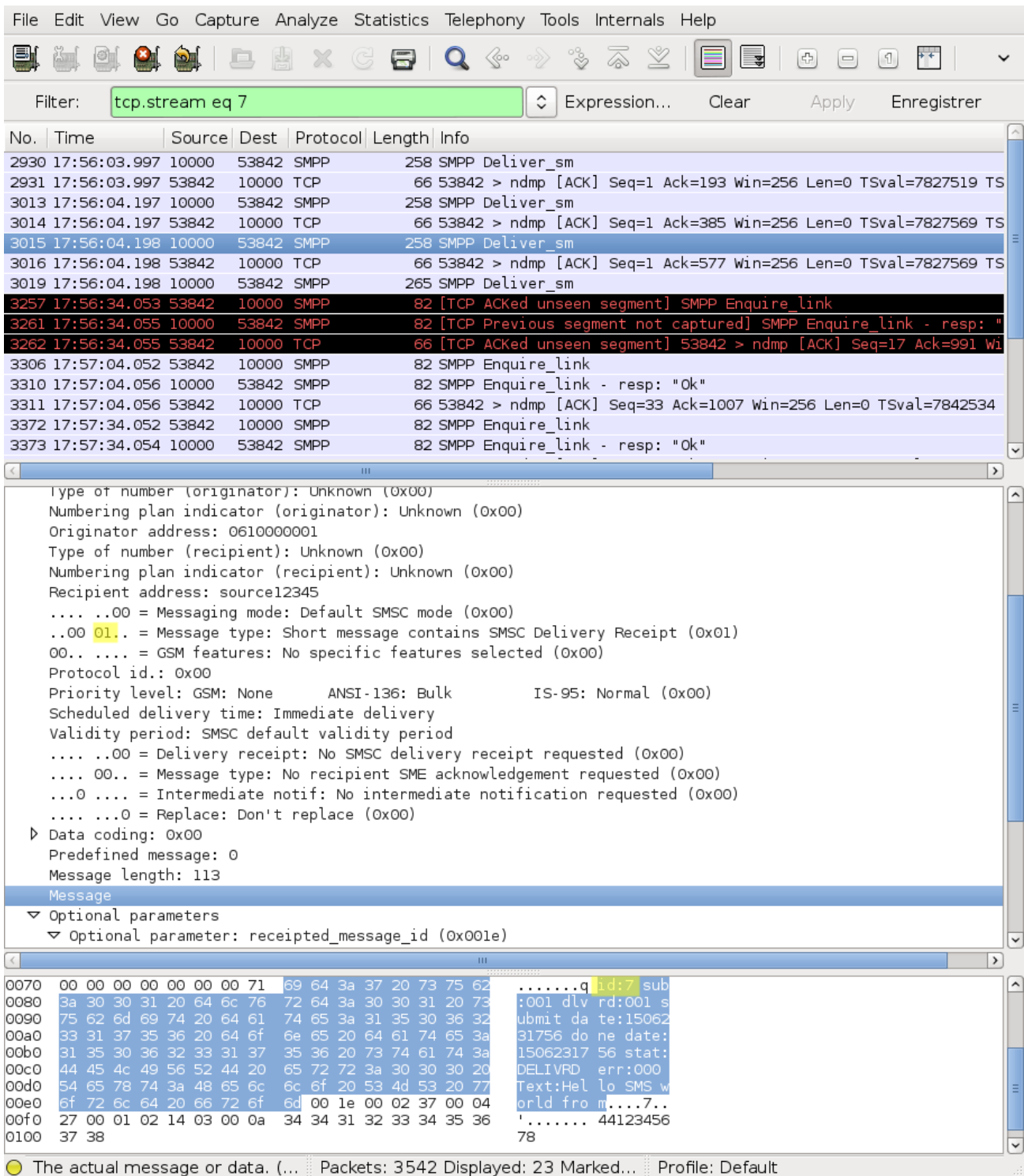
0080 30 2b 00 01 00 00 00 9f 05 00 03 67 02 01 42 65 04 .....He  
 0090 6c 6c 6f 20 77 6f 72 6c 64 2e 20 54 68 69 73 20 .....llo world. This  
 00a0 69 73 20 61 20 76 65 72 79 20 6c 6f 6e 67 20 53 .....is a very y long S  
 00b0 4d 53 20 73 70 61 6e 6e 69 6e 67 20 6d 6f 72 65 .....MS spann ing more  
 00c0 20 74 68 61 6e 20 31 36 30 20 63 68 61 72 61 63 .....than 16 0 charac  
 00d0 74 65 72 73 2e 20 41 73 20 79 6f 75 20 63 61 6e .....ters. As you can  
 00e0 20 73 65 65 2c 20 74 68 65 20 73 68 6f 72 74 11 .....see, the short.  
 00f0 6d 65 73 73 61 67 65 20 66 69 65 6c 64 20 69 73 .....message field is  
 0100 20 70 72 65 66 69 78 65 64 20 77 69 74 68 20 62 .....prefixe d with b  
 0110 69 6e 61 72 79 20 64 61 74 61 2c 20 77 68 69 63 .....inary da ta, whic  
 0120 68 20 6c 69 6d 69 74 00 05 00 01 02 00 00 00 6f .....n limit. ....o  
 0130 00 00 00 04 00 00 00 00 00 00 00 04 00 00 00 73 .....s  
 0140 6f 75 72 63 65 31 32 33 34 35 00 00 00 30 36 31 .....ourcel23 45...061  
 0150 30 30 30 30 30 30 30 00 40 00 00 00 31 35 30 36 .....0000000. @...1506  
 0160 32 39 30 37 34 36 32 35 30 30 30 2b 00 00 00 00 .....29074625 000+....  
 0170 00 74 05 00 02 66 02 02 73 20 65 61 63 68 20 70 .....\$ .....e each n



In the screenshot above, you can see the user data header in the message field (1), information contained in the UDH (2) and some extra information not belonging to the packet but computed by Wireshark (3): the *Short Message body* field is especially interesting as it contains the full message reassembled by Wireshark.

## Receiving SR

When the receiver is bound, the SMS-C may send SR at any time. The SR is sent using a DELIVER\_SM PDU with bits 2-5 of *esm\_class* set.



The DELIVER\_SM PDU must be replied to quickly by a DELIVER\_SM\_RESP PDU with the same *sequence\_number*.

To find the MT matching this SR, search for a SUBMIT\_SM\_RESP with the same ID. For example, this is the MT matching the SR

above:

Wireshark capture showing SMPP traffic. The packet list displays various SMPP packets, including Submit\_sm and Deliver\_sm. Packet 2981 is selected, showing details for an SMPP Submit\_sm response. The raw data section shows the hex and ASCII representation of the packet, with the 'short\_message' field highlighted in blue.

Identifer of the submitted sho... Packets: 3391 Displayed: 30 Marke... Profile: Default

SR are sent only if the *registered\_delivery* field is set in the MT.

The Adobe Campaign v6 SMPP connector does not handle SR that arrive before the SUBMIT\_SM\_RESP packet. The specification does not explicitly forbid this behavior, but it is considered as bad behavior (it would mean that the message has been received before it has been sent). If you encounter this case too often, ask your SMS-C partner to fix his platform.

## Deciphering the *short\_message* field of SR

The text field of the SR PDUs has a special encoding described in the *Appendix B* of the SMPP protocol specification. Unfortunately, this format is *only a recommendation* without being part of the protocol, even though most SMS-C respect more or less this very format.

You should directly ask the SMS-C partner for a documentation of its own implementation and double-check that it matches what you see in Wireshark. More often than not, SMS-C implementors don't even know their implementation and this leads to problems and misunderstandings. Do not hesitate to ask the SMS-C partner for help if there are any doubts about this field (especially the error codes).

The basic format is as follows:

```
id:IIIIIIIIII sub:SSS dlvr:DDD submit date:YYMMDDhhmm done date:YYMMDDhhmm stat:DDDDDDD err:EEE
Text:.....
```

These are general guidelines for reading the above line:

- The id is the same that has been sent in the SUBMIT\_SM\_RESP of the matching MT.
- Just ignore problems in the *text* field: this field is ignored by Campaign because it is useless, unreliable, and may even be

- completely unreadable if the SMS was sent using another encoding than pure alphanumeric ASCII. This is normal behavior.
- Field names are case insensitive (for example, id: sub: Text: can also be noted as ID: SUB: text:).
- The *dlvrd* field is generally not reliable, unless documented by the SMS-C partner.
- The dates may have any time zone, making them practically useless, or they are just plain wrong because the remote server's clock is off.
- The *stat* field may have other values than the ones defined in Appendix B. Use common sense and the SMSC partner's documentation to understand its meaning.
- The *err* field is completely dependent on the SMS-C, and most of the time documented by the SMS-C partner. Often, the code 000 means a success, while any other code indicates errors. The field is often numeric but can also be hexadecimal.

## Multiple SR for the same MT

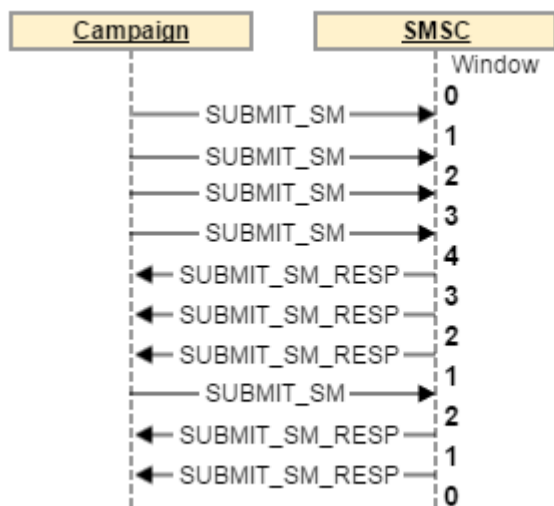
Some SMS-C send multiple SR for the same MT to track the MT's progression in the network. This is mostly useless because most of the time the client only wants to know when the message was received (this is typically the last SR).

When in doubt, only work on the latest SR received from the SMS-C to find the state of a message.

## SMPP window

Since operations and responses are asynchronous, you can optimize transfer rates by sending multiple operation PDUs before waiting for the responses. The number of messages that have no reply is called the window.

Example of a transmission with a maximum window of 4:



The current implementation does not control the window and expects the remote SMS-C to be fast enough to handle MT.