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

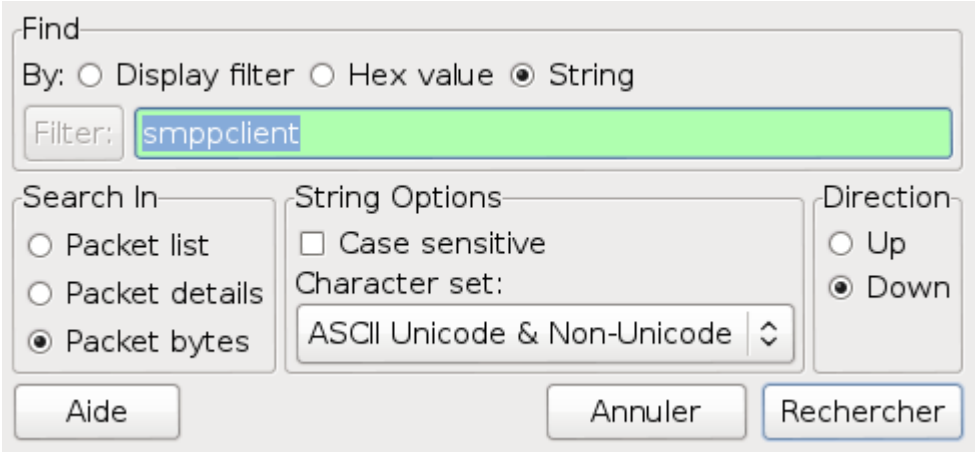
This technote assumes that you're familiar with the basics of Wireshark (capturing packets, defining simple filters, reading packet details, ...). A brief introduction is available on howtogeek - How to Use Wireshark to Capture, Filter and Inspect Packets.

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

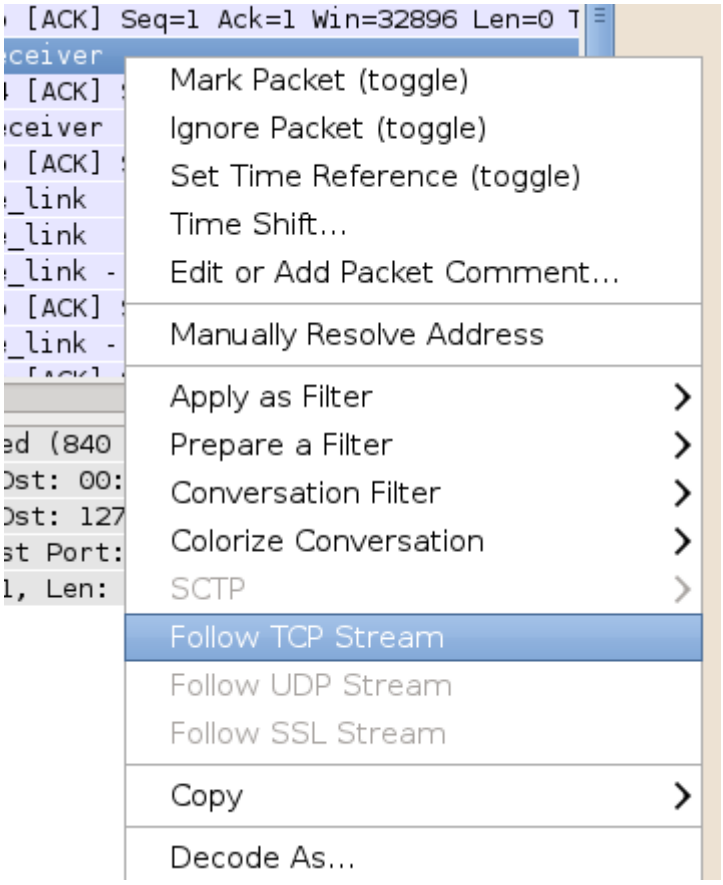
- Use a display filter on the port of the SMS-C. For example, if the SMS-C uses port 10000, use the following filter:

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

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



- 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.



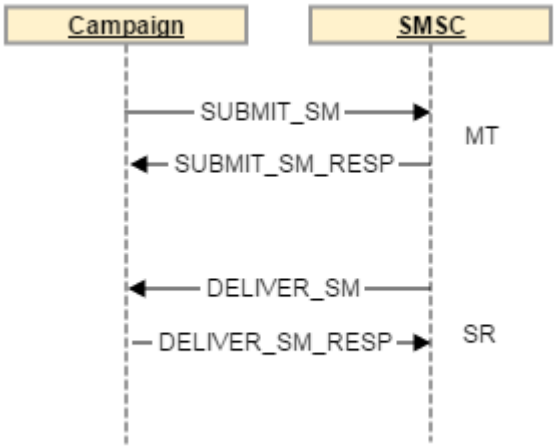
The SMPP protocol

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

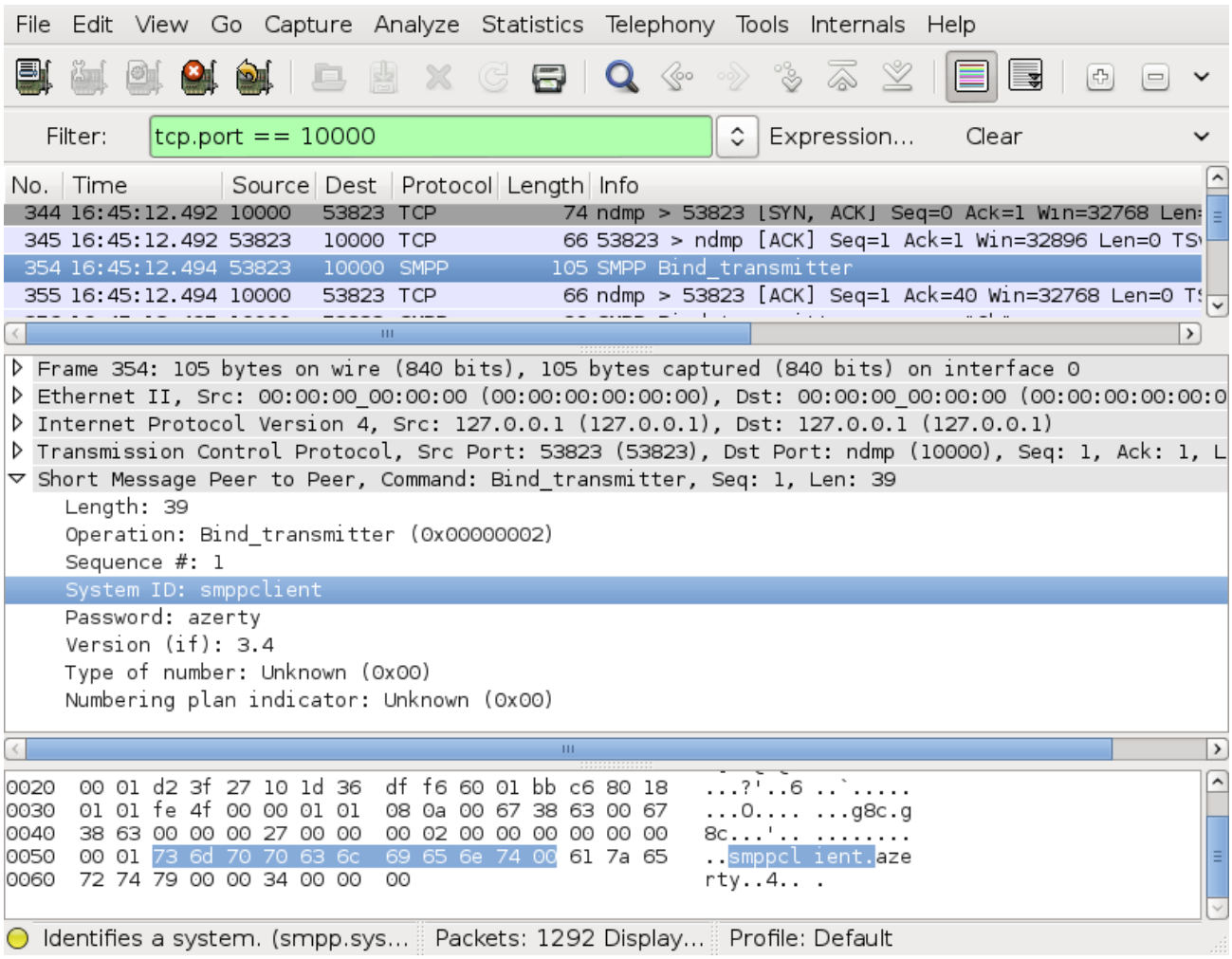
The SMPP protocol requires all commands to be acknowledged by a response PDU: BIND_TRANSMITTER is acknowledged by BIND_TRANSMITTER_RESP, SUBMIT_SM is acknowledged by SUBMIT_SM_RESP, etc.

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*).



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.

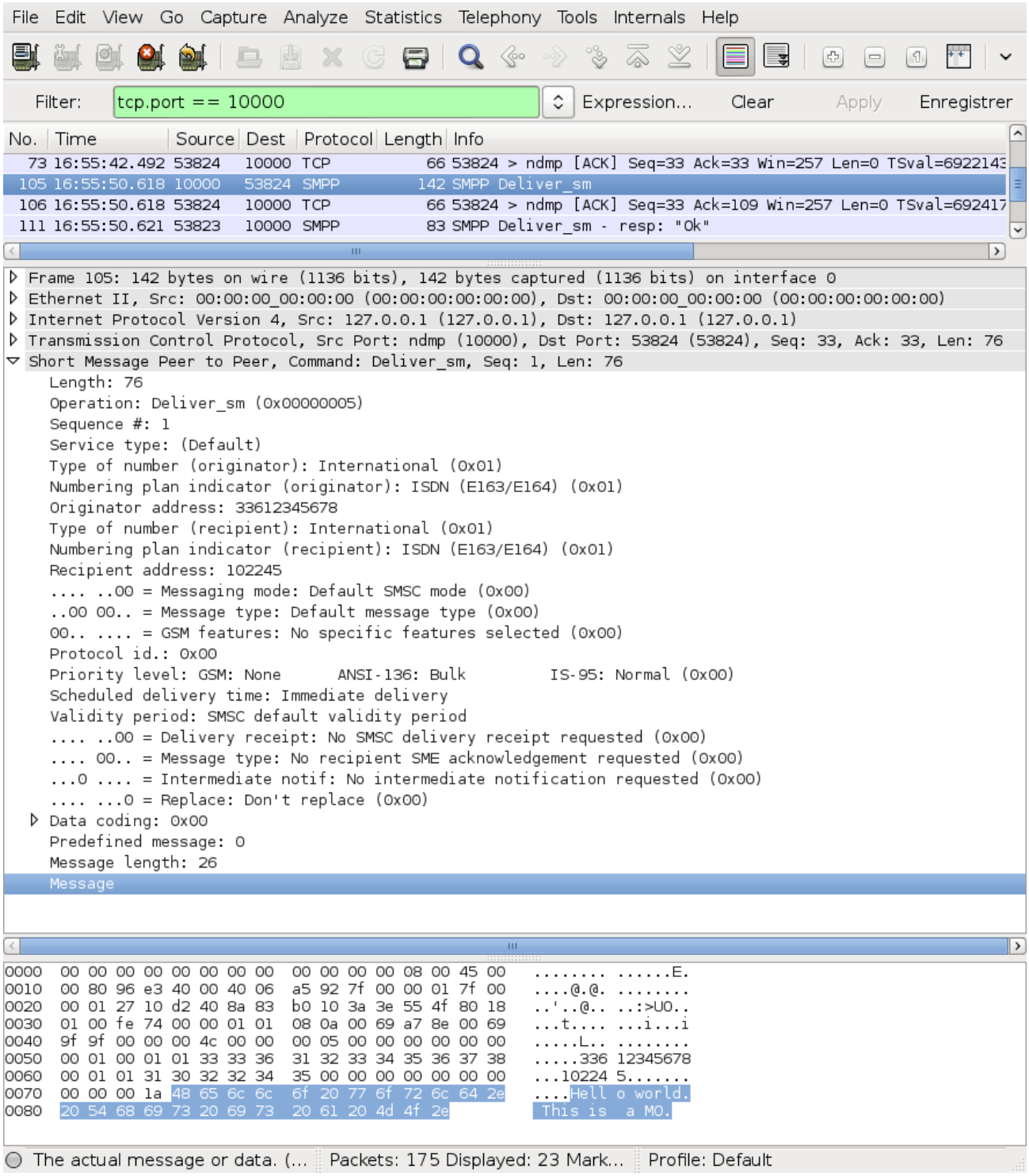
Transmitter, receiver and transceiver: The SMPP connector for Campaign v6 works in a separate transmitter/receiver mode: there are two TCP connections, one for transmitting MT and another for receiving MO and SR. Note that the TCP connection is always initiated by Campaign, even for the receiver mode.

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).



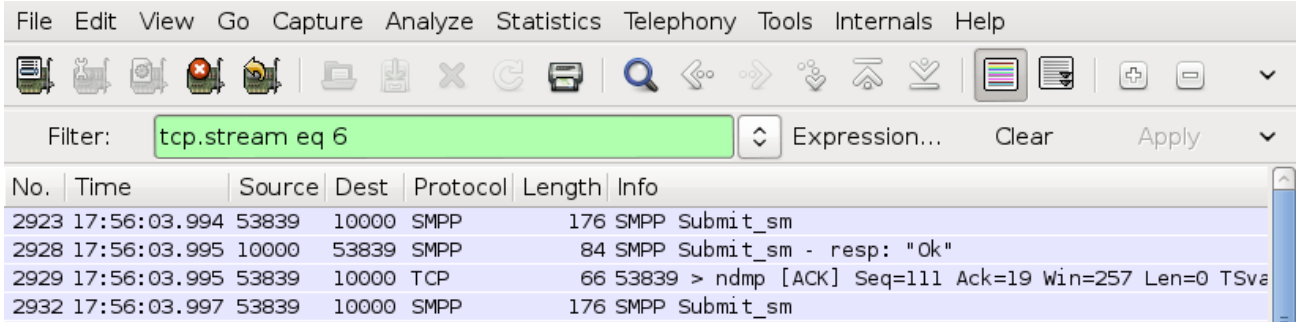
The DELIVER_SM PDU must be replied to quickly by a DELIVER_SM_RESP PDU with the same *sequence_number*.

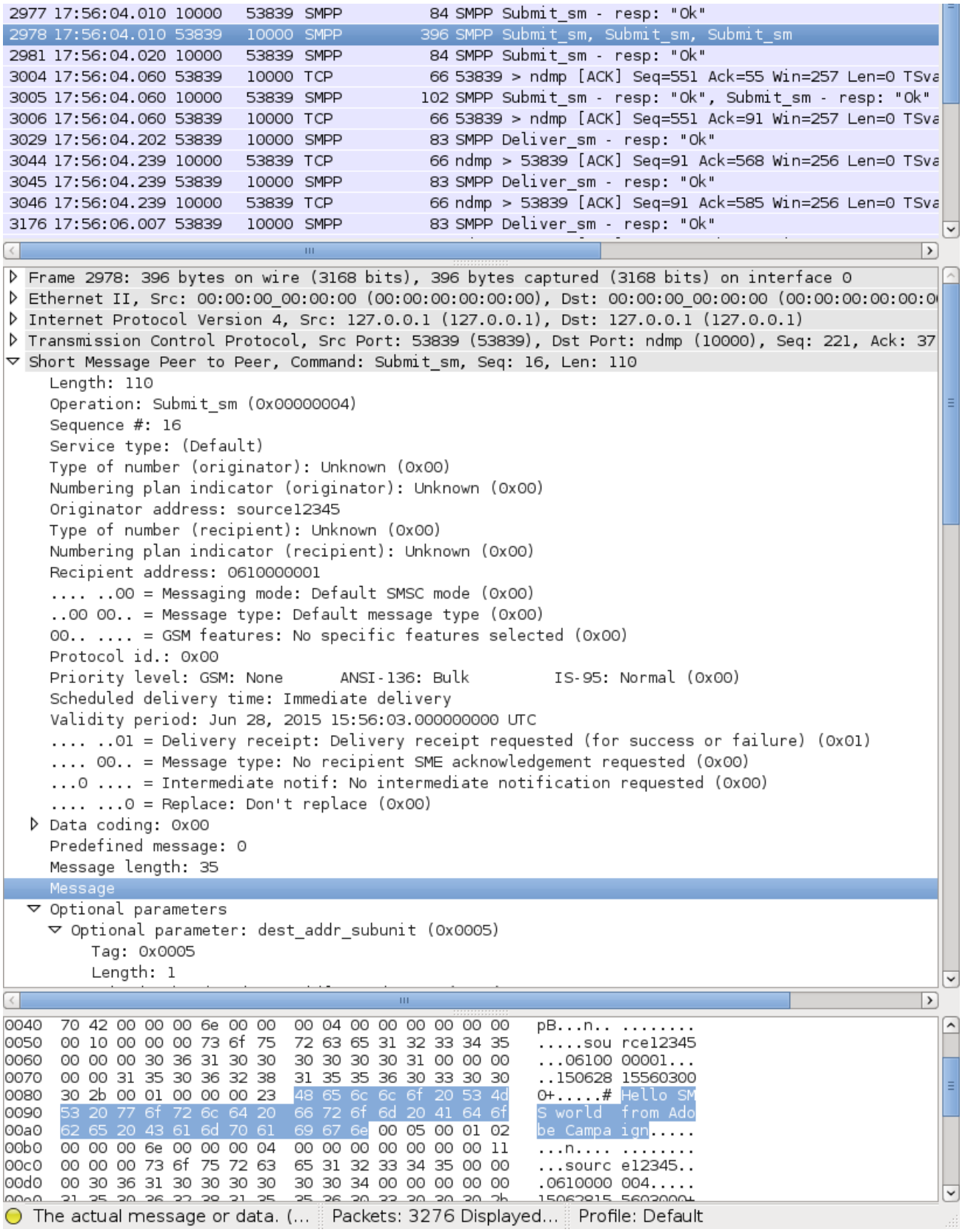
Sending MT

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.





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 *è* are part of GSM7, but *ê*, *â* or *ï* 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	GSM7 basic character set + extension (extended characters take 2 characters)
Latin-1	3	140	134	ISO-8859-1
UCS-2 UTF-16	8	70	67	Unicode (varies from phone to phone)

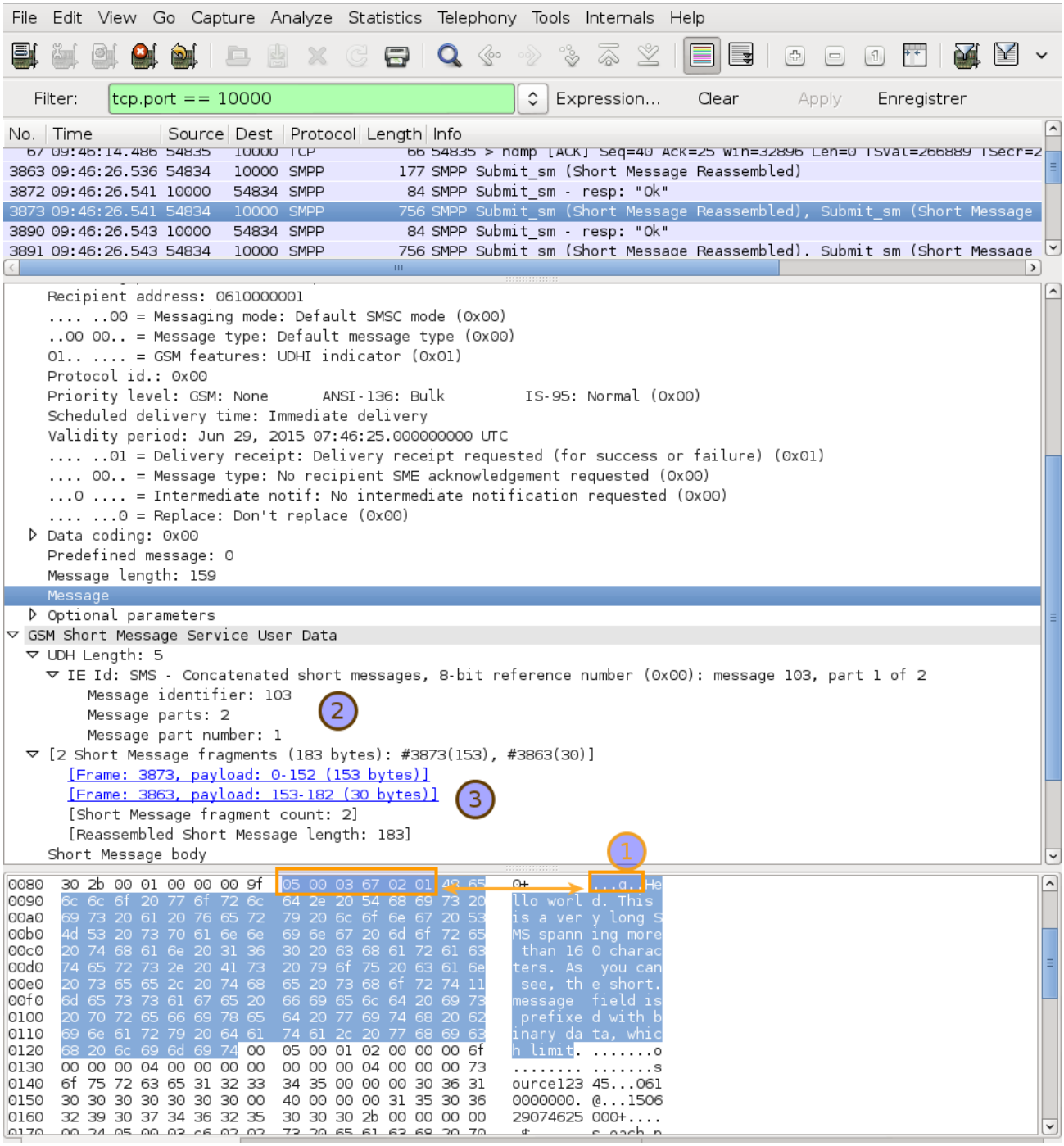
UDH

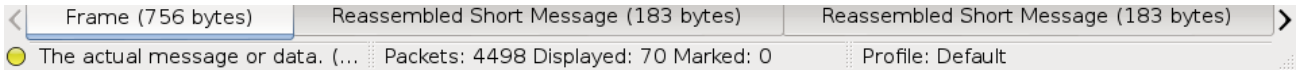
UDH (User Data Header) are small binary headers added to the text of an SMS. They can trigger special features like SMS concatenation, national language shift tables, logos/images (rarely used) or WAP push.

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.

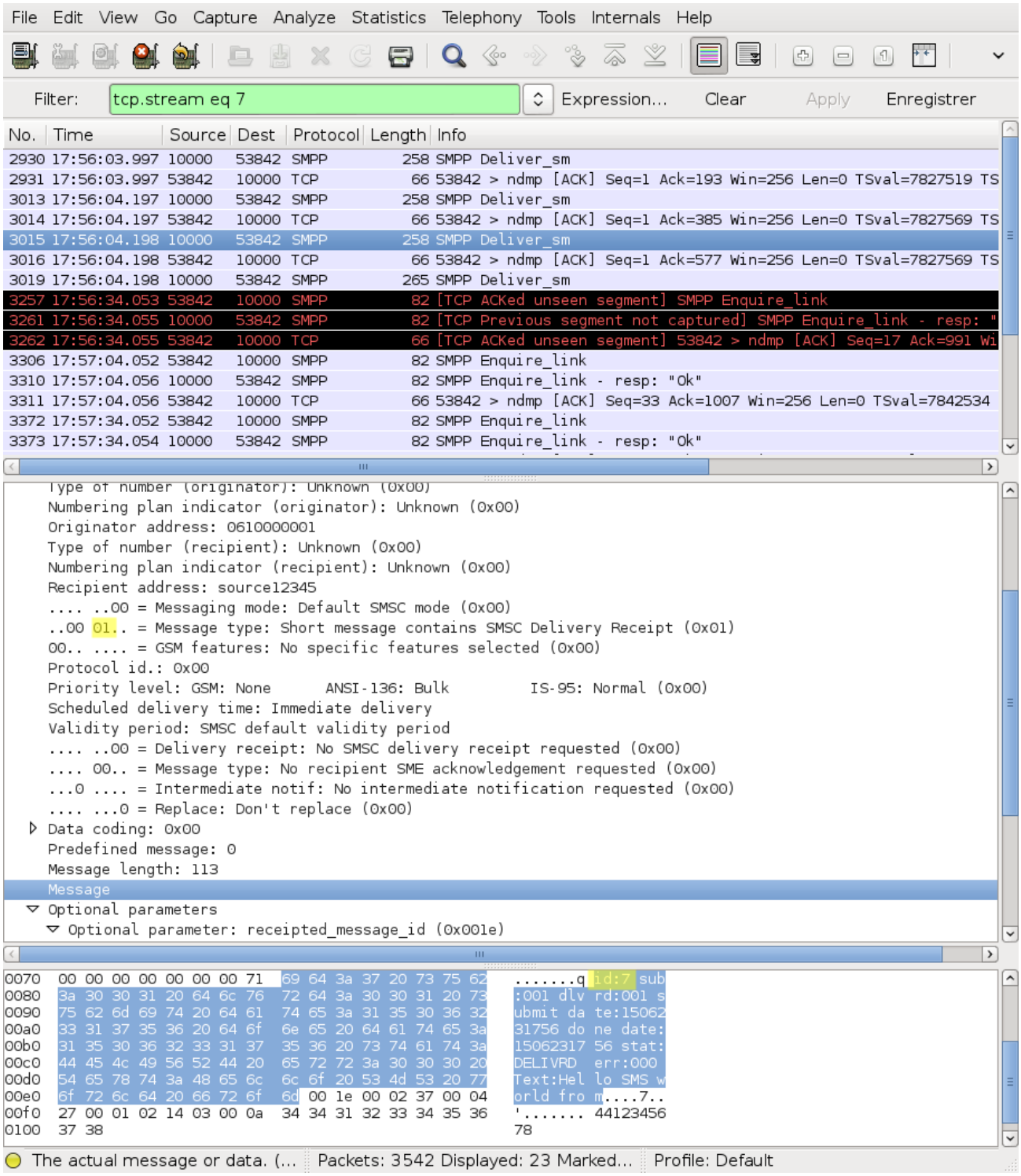




In the screenshot above, you can see the user data header in the message field (1), information contained in the UDH (2) and some extra information not belonging to the packet but computed by Wireshark (3): the *Short Message body* field is especially interesting as it contains the full message reassembled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

Receiving SR

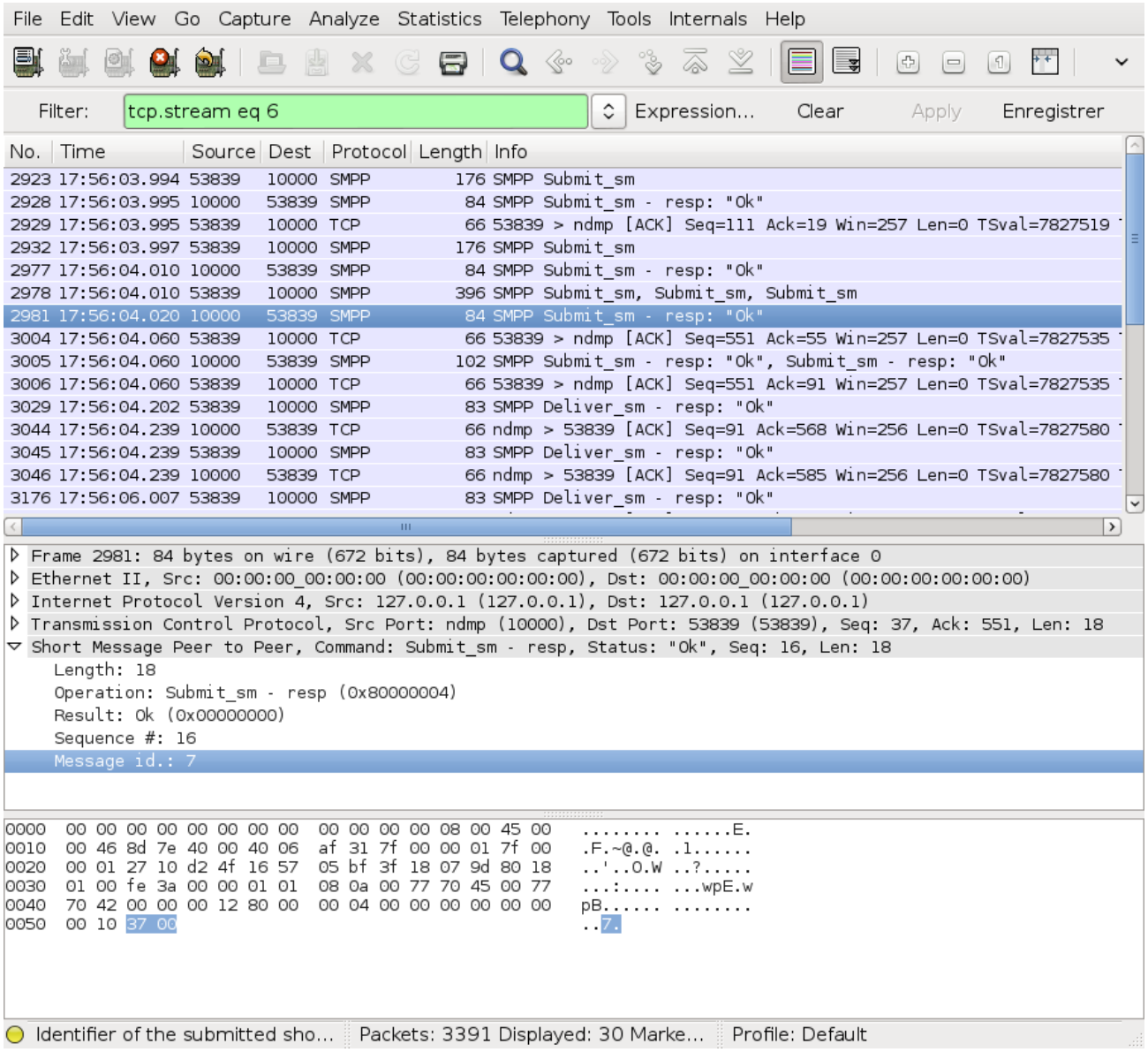
When the receiver is bound, the SMS-C may send SR at any time. The SR is sent using a DELIVER_SM PDU with bits 2-5 of *esm_class* set.



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:



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:IIIIIIIII sub:SSS dlvrd: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

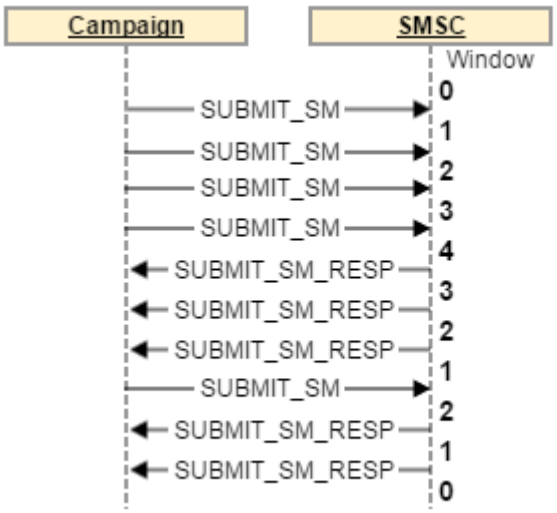
Some SMS-C send multiple SR for the same MT to track the MT's progression in the network. This is mostly useless because most of the time the client only wants to know when the message was received (this is typically the last SR).

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.