UNDERSTANDING SMS: Practitioner's Basics

Michael Harrington, CFCE, EnCE

It is estimated that in 2006, 72% of all mobile phone users world wide were active users of SMS or text messaging. In European countries like Norway, SMS usage is up around 90% of the population. The United States is quickly approaching that figure.

It should come as no surprise that youths are among the biggest users of SMS, with some students in Europe and Asia sending and receiving as many as 100 SMS messages a day and some even reaching into the 1000s per month. Seeing these figures, is it any wonder that it has been suggested that SMS text messaging is the most addictive digital service and is equivalent to a cigarette addiction.

SMS text messaging is an industry worth over 80 Billion Dollar and has had profound social, political and criminal impact. It is an important and sometimes crucial piece of electronic evidence and therefore important for the mobile forensic practitioner to understand.

This paper seeks to explain the SMS protocol and how it works across the wireless network. In addition, an example message will be encoded and decoded manually so that the practitioner can validate what his or her tool of choice is showing as the decoded message.

<u>What is SMS?</u>

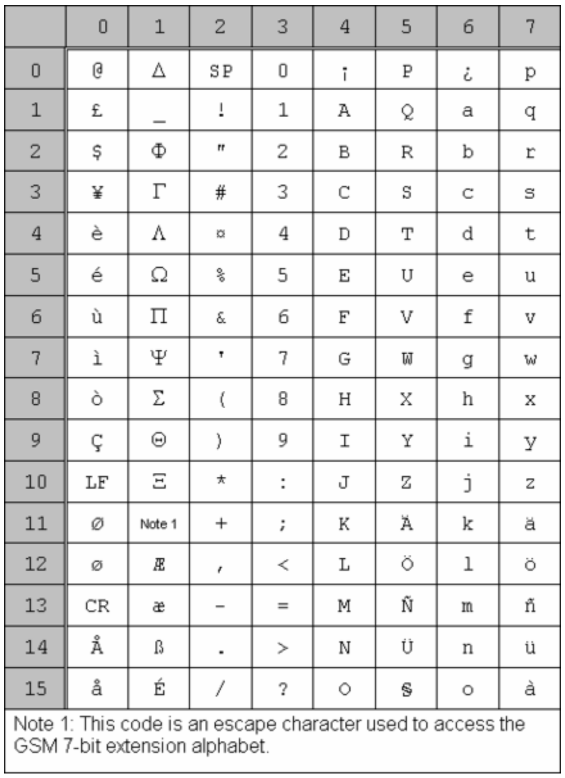
The Short Messaging Service, or SMS, is a bi-directional service to send text over wireless communication systems. It consists of a message that can be up to 160 alphanumeric characters in length. Though originally a GSM service, SMS messages are now a globally accepted service.

When a user sends a SMS text message, the message is delivered to a Short Message Service Centre or SMSC. The SMSC will attempt to deliver the message to the intended recipient. If the recipient is not available the SMSC then places the message in a que for later delivery. The SMSC supports both mobile originated and mobile terminated operations. The delivery of a SMS text message is called "best effort", in other words, there are no guarantees that a message will actually be delivered to its intended recipiant. Delays or complete loss of a message are not uncommon, particularly when sending between networks. SMS uses can request delivery reports which provide proof that a message did reach its intended destination though reports on failed SMS messages tend not to be reliable.

SMS Technical Details

As was stated previously above, when a user sends an SMS text message, it is delivered to the SMSC of the user’s wireless provider. The message is sent between the SMSC and the handset using the Mobile Application Part (MAP) of the SS7 protocol. The MAP provides an application layer to the various GSM, UMTS and GPRS nodes for intercommunication and delivery of services. The SS7 or Signalling System #7 are a set of telephony protocols widely used in Public Switched Telephone Networks (PTSN). A discussion of these protocols is beyond the scope of this whitepaper but citations to other resources are provided should the reader wish to explore these protocols further.

The SMS text message is limited to 140 Octets or 1120 bits in length. There are a few alphabets that can be used to encode the SMS text message. Shown below is the default GSM 7 Bit ASCII alphabet for encoding SMS Messages.



	0	1	2	3	4	5	6	7
0	@	Δ	SP	0	¡	P	¿	p
1	£	_	!	1	A	Q	a	q
2	$	Φ	"	2	B	R	b	r
3	¥	Γ	#	3	C	S	c	s
4	è	Λ	¤	4	D	T	d	t
5	é	Ω	%	5	E	U	e	u
6	ù	Π	&	6	F	V	f	v
7	ì	Ψ	'	7	G	W	g	w
8	ò	Σ	(	8	H	X	h	x
9	Ç	Θ	)	9	I	Y	i	y
10	LF	Ξ	*	:	J	Z	j	z
11	Ø	Note 1	+	;	K	Ä	k	ä
12	ø	Æ	,	<	L	Ö	l	ö
13	CR	æ	-	=	M	Ñ	m	ñ
14	Å	ß	.	>	N	Ü	n	ü
15	å	É	/	?	O	§	o	à

Note 1: This code is an escape character used to access the GSM 7-bit extension alphabet.

Figure 1: Default GSM 7 Bit Alphabet

SMS Text messages may also be encoded in the 8-bit data alphabet, and the 16-bit UCS2 alphabet Depending on the alphabet, maximum individual SMS sizes range from 160 7-bit characters, 140 8-bit characters, or 70 16-bit characters. It is mandatory for all GSM handsets to support the GSM 7-bit alphabet. Special characters though, like those found in languages such as Arabic, Chinese, Korean, Japanese or Russian must be encoded using the 16-bit UCS2 character encoding (Unicode).

SMS messages can also be "concatenated", where a single larger message is spread over one or more additional messages. In these cases the handset is responsible for putting the message into the proper order using data headers that are included with the segments. The data headers, since they are included with message segments cause the overall length of text segment to be less. An exploration of concatenated SMS messages is beyond the scope of this paper.

Further technical details on the SMS protocol can be found in GSM 03.40 and 03.41.

SMS Delivery

When the SMS is delivered to the SMSC it is processed. After the processing the following steps occur

- After processing the SMSC sends a request to the Home Location Register (HLR) and receives the routing information
- The SMS Center sends the message to the Mobile Switching Center (MSC).
- The MSC collects the recipient's information from the Visitor Location Register (VLR) and, sometimes, proceeds with an authentication operation
- The MSC forwards the message to a Mobile Server.
- The MSC returns the outcome of the Forward Short operation to the SMS Center
- The SMS Center reports delivery status of the short message back to the sender.

Below is a graphic taken from HowStuffWorks.com that illustrates the above.

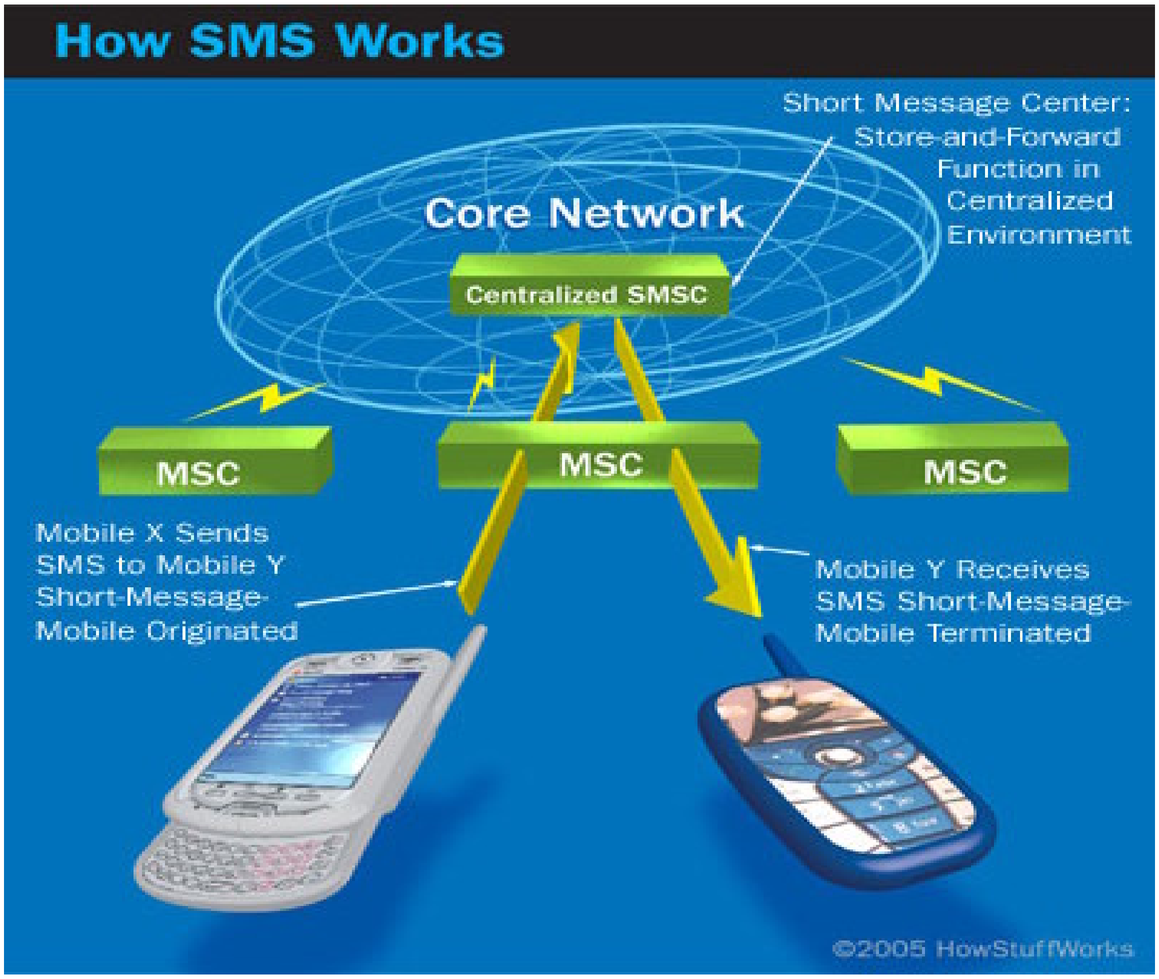

How SMS Works
Short Message Center: Store-and-Forward Function in Centralized Environment
Core Network
Centralized SMSC
MSC
MSC
MSC
Mobile X Sends SMS to Mobile Y Short-Message-Mobile Originated
Mobile Y Receives SMS Short-Message-Mobile Terminated
©2005 HowStuffWorks

Figure II: Graphic representation of SMS delivery

SMS Components

The SMS message itself is composed of several elements. Some of these elements are listed below

- Length of SMSC
- Service Center Timestamp
- Originator Address: the phone number of the sender
- Protocol Identifier
- Data Coding Scheme
- User Data Length: tells how long the message is

- User Data: the message itself (140 bytes: 160 7-bit characters, or 140 8-bit characters)

Figures III and IV taken from Dreamfabric.com show sample received and sent SMS messages.

07917283010010F5040BC87238880900F10000993092516195800AE8329BFD4697D9EC37	
Octet(s)	**Description**
07	Length of the SMSC information (in this case 7 octets)
91	Type-of-address of the SMSC. (91 means international format of the phone number)
72 83 01 00 10 F5	Service center number (in decimal semi-octets). The length of the phone number is odd (11), so a trailing F has been added to form proper octets. The phone number of this service center is "+27381000015". See below.
04	First octet of this SMS-DELIVER message.
0B	Address-Length. Length of the sender number (0B hex = 11 dec)
C8	Type-of-address of the sender number
72 38 88 09 00 F1	Sender number (decimal semi-octets), with a trailing F
00	TP-PID. Protocol identifier.
00	TP-DCS Data coding scheme
99 30 92 51 61 95 80	TP-SCTS. Time stamp (semi-octets)
0A	TP-UDL. User data length, length of message. The TP-DCS field indicated 7-bit data, so the length here is the number of septets (10). If the TP-DCS field were set to indicate 8-bit data or Unicode, the length would be the number of octets (9).
E8329BFD4697D9EC37	TP-UD. Message "hellohello" , 8-bit octets representing 7-bit data.

Figure III: Sample SMS Received message. Note: The colored cells show areas that some phones may omit.

0011000B916407281553F80000AA0AE8329BFD4697D9EC37	
Octet(s)	**Description**
00	Length of SMSC information. Here the length is 0, which means that the SMSC stored in the phone should be used. *Note: This octet is optional. On some phones this octet should be omitted! (Using the SMSC stored in phone is thus implicit)*
11	First octet of the SMS-SUBMIT message.
00	TP-Message-Reference. The "00" value here lets the phone set the message reference number itself.
0B	Address-Length. Length of phone number (11)
91	Type-of-Address. (91 indicates international format of the phone number).
6407281553F8	The phone number in semi octets (46708251358). The length of the phone number is odd (11), therefore a trailing F has been added, as if the phone number were "46708251358F". Using the unknown format (i.e. the Type-of-Address 81 instead of 91) would yield the phone number octet sequence 7080523185 (0708251358). Note that this has the length 10 (A), which is even.
00	TP-PID. Protocol identifier
00	TP-DCS. Data coding scheme.This message is coded according to the 7bit default alphabet. Having "04" instead of "00" here, would indicate that the TP-User-Data field of this message should be interpreted as 8bit rather than 7bit (used in e.g. smart messaging, OTA provisioning etc).
AA	TP-Validity-Period. "AA" means 4 days. *Note: This octet is optional, see bits 4 and 3 of the first octet*
0A	TP-User-Data-Length. Length of message. The TP-DCS field indicated 7-bit data, so the length here is the number of septets (10). If the TP-DCS field were set to 8-bit data or Unicode, the length would be the number of octets.
E8329BFD4697D9EC37	TP-User-Data. These octets represent the message "hellohello".

Figure IV: Sample SMS Sent Message

As can be seen from the above there are many elements of the SMS message. The elements of the SMS message are actually TPDU (transport protocol data units) that encapsulate the message payload. Detailed descriptions of these TDPU elements is beyond the scope of this primer but can be explored both at dreamfabric.com and in the GSM 03.40 SMS specification.

SMS and SIMs

On a Subscriber Identity Module (SIM) card, SMS messages are stored in the EF_SMS elementary file beneath the DF_TELECOM dedicated file. EF_SMS is linear fixed file meaning that the each record in the sequence of records stored there has a fixed length. The length of the SMS messages stored on the SIM card are 176 bytes in length and are identified by the header of x/6F x/3C.

The first byte of the SMS record is the status byte. The status byte takes the following values in binary

- 00000000-Unused
- 00000001-Mobile equipment terminated, read
- 00000011-Mobile equipment terminated, not read
- 00000101-Mobile equipment originated, sent
- 00000111- Mobile equipment originated, not sent

Two things are notable to the examiner (besides the message itself) in regard to SMS messages on SIM cards. First, when and SMS message is "deleted", only its status byte is set to x/00-the record retains its data until it is overwritten by another message.

The second thing of note is that there is no slack space in the records where an examiner may recover a partial message. When the previous message is overwritten by another message that does not completely take up the full space allotted, the remainder of the record is written over with x/FF. Therefore it behooves the examiner to take extreme care to prevent other messages from being sent that may potentially wipe out deleted messages.

Figure V below shows an actual SMS record from a Cingular SIM card. Using what we have learned in the previous sections we will break down the SMS record.

OFFSET	HEX DATA	ASCII
00000000	01 07 91 31 21 13 94 68 F0 24 0B A1 31 63 08 53	..•1!.•hð$.¡1c.S
00000010	44 F5 00 00 40 60 51 10 95 83 69 81 C8 B0 BD 0C	Dõ..@`Q.••i•È°½.
00000020	32 D7 DD A0 FB 0B 44 47 97 41 E7 B4 9C 3D 07 85	2×Ý û.DG•Aç´•=.•
00000030	DD 64 50 DA 0D 32 B3 C3 2D 50 FE 5D 07 AD 9D EF	ÝdPÚ.2³Ã-Pþ].••ï
00000040	3B 28 CD 7E DB CB A0 FC BB 0E 0A B3 EF E1 FC 1C	;(Í~ÛË ü»..³ïáü.
00000050	D4 4A 83 EE 61 37 1D 94 7F D7 41 F4 37 68 ED 7E	ÔJ•îa7.••×Aô7hí~
00000060	DF 41 E9 36 68 2E CF 83 C2 6E 32 28 CD 66 83 DA	ßAé6h.Ï•Ân2(Íf•Ú
00000070	E9 F9 1C 94 7F D7 41 2B D0 5C 9E 07 A5 DB A0 75	éù.••×A+Ð\•.¥Û u
00000080	DA 4D 06 85 41 E6 FA 78 5D 26 83 EA 70 FF FF FF	ÚM.•Aæúx]&•êpÿÿÿ
00000090	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000A0	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ

Figure V: SMS Record from a Cingular SIM card

01 **Message Status (mobile terminated and read**)
07 **Length of SMSC (includes Identifier)**
91 **Type of Address (International)**
3121139468F0 **SMSC (Reverse Nibbled)**
24 **Start of the SMS-Deliver**
0B **Length of sender address (Decimal 11)**
A1 **Type of sender addresss (National E.164 ISDN)**
3163085344F5 **Sender Number (Reverse Nibbled)**
00 **Protocol ID (mobile to mobile)**
00 **Encoding Scheme (Default 7 Bit GSM and immediate display)**
40605110958369 **(Time Stamp Reverse Nibbled)**
81 **(SMS Message length. This is the *HEX* representation of the character length- in this particular instance 129 characters. See below for further explanation.)**

<u>PDU Data</u>

```
C8B0BD0C32D7DDA0FB0B44479741E7B49C3D0785DD6450DA0D32B3C32D50FE
5D07AD9DEF3B28CD7EDBCBA0FCBB0E0AB3EFE1FC1CD44A83EE61371D947F
D741F43768ED7EDF41E936682ECF83C26E3228CD6683DAE9F91C947FD7412BD0
5C9E07A5DBA075DA4D068541E6FA785D2683EA70FFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Two important concepts have been mentioned out in the above breakdown. The first is the concept of reverse nibbling. This is when the individual nibbles of a byte need to be swapped in order for the data to be decoded properly e.g. 41h becomes 14h.

The second concept comes with the calculation of the length of the PDU encoded message. As was mentioned above, the length byte indicates the character length of the message. In order to decode the PDU we need to know how many bytes to carve.

The first step in determining the byte length of the message involves converting the length byte from hex to decimal.

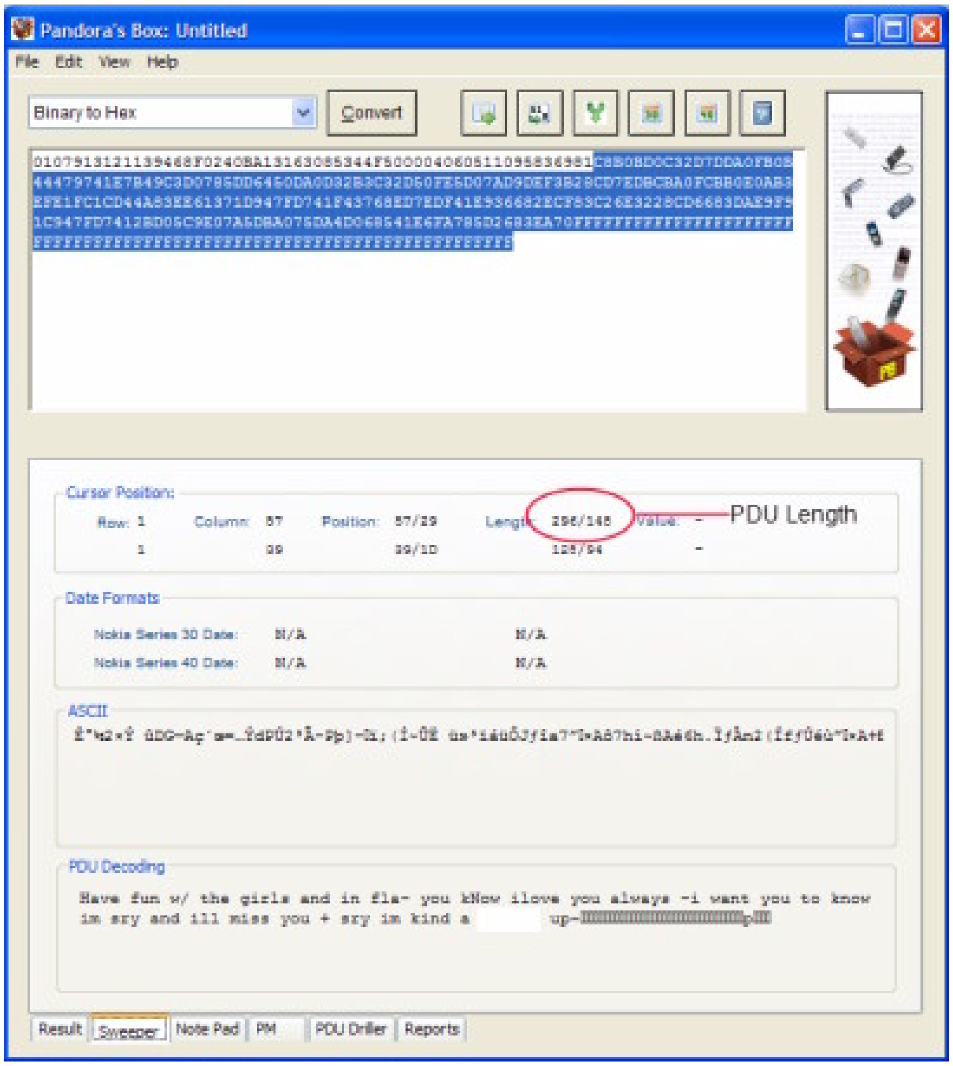
81h = 129

Then this number is divided by eight (eight bit) and the result of this division is multiplied by seven (the result is rounded up to the next whole integer). Why multiplied by seven? To account for default GSM 7 bit encoding. The method of encoding will be explored further later on in the paper.

129/8 = 16.125
16.125 x 7 = 112.87 or 113

This is the number of bytes containing our PDU-encoded message. It is important to note that in this example, our result is not a multiple of seven - so it must be padded to the next multiple of seven to again account for the seven bit encoding. If we do not perform this task then there is a danger that the result of decoding the selected data could be corrupted, in the last part of the message. The next multiple of 7 from 113 is 119, so we pad the remaining six bytes with FF which is the default for the GSM Standard (Nokia uses 00h). In the example below, you may notice that there are more FF bytes than the six I just mentioned. This because the record slot on the SIM must be filled (176 bytes).

Confusing? Well thankfully a tool like Pandora's Box allows you to sweep the hex of an SMS message and do the calculating for you (as shown in Figure VI below)



```
0107913121139468F0240BA13163085344F500004060511095836981C8B08D0C32D7DDA0FB0B
44479741E7B49C3D0786DD6450DA0D82B8C32D60FE6D07AD9DEF3B28CD7EDBCBA0FCBB0E0AB3
EFE1FC1CD44A83EE61371D947FD741F43768ED7EDF41E936682ECF83C26E3228CD6683DAE9F5
1C947FD7412BD05C9E07A5DBA075DA4D068541E6FA785D2683EA70FFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Cursor Position:

```
Row: 1     Column: 57     Position: 57/29     Length: 296/148     Value: -
     1             39               39/1D             125/94            -
```

Date Formats

```
Nokia Series 30 Date:   N/A          N/A
Nokia Series 40 Date:   N/A          N/A
```

ASCII

```
Ê"¾2×Ÿ ûDG—Aç`ø=…ŸdÞÛ2'Ã-Øp)-ñ;(Î-ÛË ùs'iäùÕJƒîa7"Ï×Aö7hi-ßAédh.ÏƒÅn2(ÎfƒÚéù"Ï×A+E
```

PDU Decoding

```
Have fun w/ the girls and in fla- you kNow ilove you always -i want you to know
im sry and ill miss you + sry im kind a [redacted] up-ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿpÿÿÿ
```

Figure VI: PDU Length Pandora's Box (offensive language redacted at the end of the message.)

SMS Encoding

As was stated at the beginning of the white paper we are going to encode and decode a SMS text message.

Before we actually proceed to encode and decode the message, the reader needs to be cognizant of a couple of assumptions. The first assumption is that we will be using the standard GSM 7 Bit ASCII alphabet. The second assumption is that we will be encoding the message using PDU encoding. Now with these assumptions out of the way let us begin to encode a text message.

Out text message we are going to encode is the simple eight character phrase (without the quotes) "SMS Rulz". Let's place the phrase into a table and number the characters

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**

Our very first step to encoding our message into PDU format is to take the hexadecimal equivalents of the ASCII text. This is shown in the next table.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**

Now that we have our hex values for each ASCII letter of our message we need to convert those values into their binary equivalents. Why? So we can begin the process of compressing our 8 bit message into 7 bits. This will involve some bit swapping.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
01010011	**01001101**	**01010011**	**00100000**	**01010010**	**01110101**	**01101100**	**01111010**

In order to start encoding our SMS message we need to take the ***most** significant bit* of each octet and discard it. The most significant bit is shown in red below.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
01010011	**01001101**	**01010011**	**00100000**	**01010010**	**01110101**	**01101100**	**01111010**

With the most significant bit of each octet discarded, we have effectively completed the compression portion of this conversion process. The remaining eight *septets* (8 x 7 bits = 56 bits) must be converted into seven *octets (7 x 8 bits = 56 bits)*.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**U**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
1010011	**1001101**	**1010011**	**0100000**	**1010010**	**1110101**	**1101100**	**1111010**

Now to start the conversion from eight septets to seven octets we need to take the **least** significant bit from septet number two and place it as the **most** significant bit on septet number one. This is shown below in red and green. Septet one now grows by one bit and becomes our first octet.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**U**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
11010011	**100110x**	**1010011**	**0100000**	**1010010**	**1110101**	**1101100**	**1111010**

But now we are faced with a two bit deficit on septet number two. In order to make this a full octet we must take the two **least** significant bits from septet number three and add them as the two **most** significant bits of septet number two. Again, this is shown in red and green below. We now have our second octet.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
11010011	**11100110**	**10100xx**	**0100000**	**1010010**	**1110101**	**1101100**	**1111010**

Now, as a result of the last process, as can be seen below septet number three is missing three bits which must be made up from the last three **least** significant bits of septet four.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
11010011	**11100110**	**00010100**	**0100xxx**	**1010010**	**1110101**	**1101100**	**1111010**

As you can by now no doubt guess this process of moving the appropriate **least** significant bits of the following septet to the **most** significant position of the previous septet is repeated for each of the remaining septets. We take one more bit each time, enough to make up the deficit required to complete the octet. This is shown in the following tables.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
11010011	**11100110**	**00010100**	**00100100**	**101xxxx**	**1110101**	**1101100**	**1111010**

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
11010011	**11100110**	**00010100**	**00100100**	**10101101**	**11xxxxx**	**1101100**	**1111010**

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
11010011	**11100110**	**00010100**	**00100100**	**10101101**	**10110011**	**1xxxxxx**	**1111010**

Note that we use all the remaining bits of the 8th septet to fill the seven **most** significant missing bits of our seventh Octet.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
11010011	**11100110**	**00010100**	**00100100**	**10101101**	**10110011**	**11110101**	**xxxxxxx**

We are now left with seven octets from our eight septets! Our final step involves converting our binary numbers to their hexadecimal equivalents.

D3	**E6**	**14**	**24**	**AD**	**B3**	**F5**
11010011	**11100110**	**00010100**	**00100100**	**10101101**	**10110011**	**11110101**

So our message “SMS Rulz” changed from

53 4D 53 20 52 75 6C 7A

To the PDU Encoded

D3 E6 14 24 AD B3 F5

SMS Decoding

Now that we know how to encode the message, let’s take the process in reverse and decode it. This will help us validate our encoding and also illustrate the process automated tools like Pandora’s Box use for the decoding process.

From above we have our encoded hexadecimal string

D3 E6 14 24 AD B3 F5

We need to take these values and get their binary equivalents in order to expand out our seven septets into eight.

1	2	3	4	5	6	7
D3	E6	14	24	AD	B3	F5
11010011	11100110	00010100	00100100	10101101	10110011	11110101

Now we need to go in the reverse of our previous discussion, creating eight septets from our seven octets. To do this we take the seven **most** significant bits of Octet seven and place them into an eighth septet as shown in red and green below.

1	2	3	4	5	6	7	8
11010011	11100110	00010100	00100100	10101101	10110011	xxxxxxx1	1111010

But now of course we now need to create septet number seven. So we take the six **most** significant bits from octet six and place them as the **least** significant bits of septet seven.

1	2	3	4	5	6	7	8
11010011	11100110	00010100	00100100	10101101	xxxxxx11	1101100	1111010

And now, of course, the five **most** significant bits go to the **least** significant bits of septet six.

1	2	3	4	5	6	7	8
11010011	11100110	00010100	00100100	xxxxx101	1110101	1101100	1111010

As you may have already surmised this process follows down the line for each remaining septet, transporting one less bit each time.

1	2	3	4	5	6	7	8
11010011	11100110	00010100	xxxx0100	1010010	1110101	1101100	1111010

1	2	3	4	5	6	7	8
11010011	11100110	xxx10100	0100000	1010010	1110101	1101100	1111010

1	2	3	4	5	6	7	8
11010011	xx100110	1010011	0100000	1010010	1110101	1101100	1111010

1	2	3	4	5	6	7	8
1010011	1001101	1010011	0100000	1010010	1110101	1101100	1111010

We are now left with eight septets. But wait … Don’t we need eight octets to reconstitute our original message? Indeed! Recall that when we started out PDU encoding we removed the least significant bit from each of the eight octets (which was a zero).

To get us back to our original string we need to add that least significant bit back onto each octet (which is a zero). This results in the table below.

1	2	3	4	5	6	7	8
01010011	**01001101**	**01010011**	**00100000**	**01010010**	**01110101**	**01101100**	**01111010**

We then get the hex values and their ASCII equivalents and end up back at our original un-encoded message!

1	2	3	4	5	6	7	8
01010011	**01001101**	**01010011**	**00100000**	**01010010**	**01110101**	**01101100**	**01111010**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
S	**M**	**S**		**R**	**u**	**l**	**z**

Conclusion

The global SMS industry is worth over 80 billion dollars in annual revenue and shows no sign of shrinking. SMS text messaging is a rapidly growing method of communication that can be used for all types of criminal activity. It behooves the examiner to understand the components of an SMS messages and how the SMS messages interact and are delivered in the wireless network.

This paper examined the components of the SMS text message, how it is delivered and how text messages are encoded for delivery. It is my sincere hope that it will be of benefit to the mobile forensic community.

Acknowledgements

The writing of this white paper would not be possible without help. I would like to thank DC Stephen Hirst (especially with the encoding/decoding!) and DC Stephen Miller for their continued help and guidance and sharing of knowledge. Thanks also to Ian Holliland of Leicestershire Constabulary, DC Carole Walton and to Det. Brian Roach for their critical eye and suggestions. And, of course, thanks to all the fine practitioners at phone-forensics.com!

References

1. http://en.wikipedia.org/wiki/SMS
2. http://en.wikipedia.org/wiki/Mobile_Application_Part
3. http://en.wikipedia.org/wiki/Signaling_System_7
4. http://www.dreamfabric.com/sms/
5. http://communication.howstuffworks.com/sms.htm
6. Jurgensen, Timothy M., Scott B. Guthery, *Smart Cards: The Developer's Toolkit*, 2002
7. GSM 03.38 and GSM 03.40