Introduction to SMS

Presented by developerWorks, your source for great tutorials

ibm.com/developerWorks

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

Section 1. Introduction to the tutorial

Who should take this tutorial?

This tutorial gives the reader a high-level introduction to Short Messaging Service (SMS). The course is intended for developers and technical managers who want to get an overview of SMS.

About this tutorial

This tutorial provides an introduction to basic SMS (short messaging service) concepts, specifications, high-level architecture, and examples of SMS applications. SMS is a wireless service that delivers alphanumeric messages to mobile phones. Within the last few years SMS has become a popular mode of transmitting short text messages. SMS application development is fairly simple and is designed to work with standard Internet protocols such as SMTP and HTTP.

Author Vivek Malhotra is the technical director of the Wireless Solutions Practice at *Etensity, Inc.* , a technology and professional services company focusing on strategy, technology development, systems engineering, and managed application services. Vivek has several years of experience developing and implementing wireless applications. He has spoken on expert panels focusing on the wireless industry. For any questions about the content of this tutorial, contact the author at *vmalhotra@etensity.com* .

Prerequisites

You should be familiar with the HTTP protocol, basic Active Server Pages (ASP) and basic Java Server Pages (JSP) development.

Section 2. SMS overview

Definition of SMS

SMS is the transmission of alphanumeric messages to and from mobile phones to external systems like e-mail and pagers.

Overview and history

SMS is the delivery of alphanumeric messages to mobile phones over wireless networks. The length of the message can be no longer than 160 characters. In Europe, two-way SMS messaging has been popular for some time and is slowly gaining popularity in North America as some of the major wireless networks (like AT&T) are beginning to support it. SMS is a universal data service and is supported on GSM, TDMA, and CDMA networks. An SMS message can originate from an external system such as e-mail or mobile device and is routed through the network, via the short messaging service center (SMSC), to its destination. A distinguishing feature of SMS is its ability to deliver messages any time, regardless of whether data or voice calls are in progress.

Benefits of SMS

SMS first appeared in Europe in 1991 as part of the Global System for Mobile Communications (GSM) Phase 1 standard. SMS was made available in North America recently, and was first adopted on digital networks built by early wireless carriers such as BellSouth Mobility, Nextel, and AT&T. SMS is supported on digital wireless networks based on GSM, code division multiple access (CDMA), and time division multiple access (TDMA). SMS has a number of benefits, which include:

* Guaranteed delivery of notifications and alerts to single or multiple users
* Increased user productivity through instant delivery of notifications and alerts
* Low cost and reliable communication mechanism for information delivery
* Integration with Internet-based applications
* Another service and source of revenue for service providers
* Very possible replacement of existing two-way paging

Section 3. Architecture

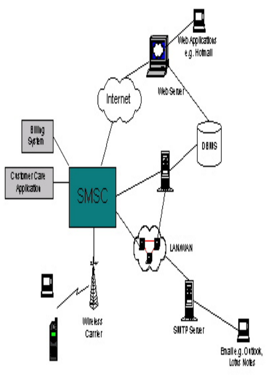
Basic elements within SMS infrastructure

The next panel describes the elements illustrated in this diagram.

SMS architecture description

The following are descriptions of the SMS architecture elements:

* **MS** -- Mobile station, a wireless terminal that is capable of receiving and sending alphanumeric messages.
* **SME** -- Short message entity, which can be a device like a mobile phone, or an application like e-mail that is capable of receiving and sending alphanumeric messages.
* **SMSC** -- Short message service center, responsible for storing and forwarding messages to and from the mobile station.
* **STP** -- Single transfer point, which allows for interconnections over signaling system 7 (SS7) links and multiple network elements. For more information on SS7 links, see the Resources section.
* **HLR** -- A database in the network that holds information like subscriber and service profile, as well as subscriber. Routing information for the subscriber is also stored in the HLR, which is requested by the SMSC.
* **MSC** -- Mobile switching service center. The job of the mobile switching service center is to switch connections between mobile stations, or between mobile stations and the fixed network.
* **BS** -- Base station, which relays information to and from the mobile station to the MSC. The BS consists of controllers and transceiver stations, also known as "cells."
* **MS** -- Mobile station, a wireless terminal that is capable of receiving and sending alphanumeric messages.


Web Applications e.g. Hotmail
Web Server
Internet
Billing System
DBMS
Customer Care Application
SMSC
LAN/WAN
Wireless Carrier
SMTP Server
Email e.g. Outlook, Lotus Notes

SMS network infrastructure

This figure illustrates how SMS can be integrated with applications over the Internet and applications within a LAN/WAN/Intranet. Integrating applications like notification services (voice/fax, reminder, calendar, etc.), e-mail, information services (weather alerts, stock alerts, etc.), and WAP integration are some examples of easily integrating SMS.

Section 4. SMS applications

Consumer applications

Examples of SMS consumer applications include:

* **Peer-to-peer messaging:** This is the most common type of use of SMS. A message like, "Hello, how are you?" or "Meet you at 8:00 PM for dinner" is usually exchanged between two mobile users. This is a quick, efficient, and inexpensive method of communication.
* **Information services:** Information services include stock quotes, weather forecasts, and news updates. A simple SMS application would require a user to type in "ST" for stock quotes or "WEA" for weather forecast. Upon submitting the request, the user receives the appropriate information in the form of an SMS message.
* **Advertising:** SMS can be used to send targeted alerts to a user. The user would sign up to receive special alerts informing the user of upcoming events. Additionally, businesses can use SMS as a form of low-cost advertising.

Commercial/enterprise applications

Examples of SMS commercial/enterprise applications:

* **Customer service:** SMS can be used as a customer service tool, thereby avoiding expensive person-to-person voice calls to customer service centers. This is an efficient and inexpensive method of providing account status and other pertinent information.
* **Job dispatch:** SMS can be used in job dispatching applications to communicate information between office-based and mobile staff. Sending an address to a message courier in the field is one example of a job dispatch application. The dispatch application can be integrated with other applications, such as vehicle positioning applications.

Section 5. Sending a short text message

Message application

Writing an SMS application is fairly simple. Carriers have made it difficult, however, for developers to write SMS applications because the carriers would have to expose their APIs, making them available over TCP/IP, which they are reluctant to do. However, most of the carriers have exposed Simple Mail Transport Protocol (SMTP), which allows developers to write short text messages through an e-mail interface. This section walks you through the implementation of an application to send a short text message to a mobile phone. Note, that for the example to work, you will have to have access to a valid SMTP server.

Carrier and domain name

Before sending a message to a mobile phone, you must know the carrier's domain. Typically, the e-mail address of the mobile user is the user's phone number followed by the carrier's domain. For example, a message sent to an AT&T phone will have an e-mail address of 18005551212@mobile.att.net. Below is a list of some of the major carriers and their domains:

* AT&T: mobile.att.net
* Sprint: messaging.sprintpcs.com
* Nextel: messaging.nextel.com
* Verizon: mst.myVZw.com
* Voicestream: voicestream.net

Length of the message

Typically, the length of a short text message is 160 characters. However, most carriers set their own limitations. The table below gives the maximum length of a message that is supported by the carriers.

Carrier	**Length of Message**
AT&T	140
Nextel	280
Sprint	100
Verizon	120
Voicestream	140

Short text message via SMTP -- An example-using ASP

The outline of the application is as follows:

```
 <%@ Language="VBScript">
  <%
  'Variable declaration
  .
  .
  'Retrieve message information
  .
  .
  'Assign carrier lengths
  .
  .
  'Create instance of message
  .
  .
  'Build the Message
  .
  .
  'Send the message
  .
  .
  %>
```

Variable declaration

We'll be using the following variable declaration for the short text messaging implementation:

```
dim msgFrom 'The person sending the message dim msgTo 'To whom
the message is being sent dim msgCarrier 'Name of carrier dim
msgText 'The actual message being sent
```

Retrieve message information

Typically, the message being sent will have the e-mail address of the user, from whom the message is being sent, as well as the message body. The subject field is not included because it would take up additional characters that could be used by the message body. When sending a message, its total length is the sum of the from, to, and message text fields (and subject field if included). The following shows how to retrieve the message information from the form submitting the message.

```
msgFrom = Request("from") msgTo = Request("to") msgCarrier =
Request("carrier") msgTxt = Request("message")
```

Create message instance

For this example, we use the CDO for NTS ("Cdonts") object for Windows 2000 Server to handle sending the message and communicating with the SMTP server. Other objects, such as DevMailer 1.0, ASP, and ASPEmail 4.5 could be used instead. More on DevMailer can be found at *Geocel International* and ASPEmail can be found at *Persits Software, Inc.* The following statement creates an instance of the message using Cdonts:

```
Set sendmail = Server.createObject("CDONTS.NewMail")
```

Assign carrier message length

The length of the message is based on the carrier network the message is being sent to. The length must be set, as different carriers have determined what lengths they can accommodate.

```
' AT&T
If (msgCarrier = "ATT") Then
maxmsgLength = 140
msgDomain = "mobile.att.net"
' Nextel
ElseIf (msgCarrier = "Nextel") Then
maxmsgLength = 280
msgDomain = "messaging.nextel.com"
' Sprint PCS
ElseIf (msgCarrier = "Sprint") Then
maxmsgLength = 100
msgDomain = "messaging.sprintpcs.com"
'Verizon
ElseIf (msgCarrier = "Verizon") Then
maxmsgLength = 120
msgDomain = "msg.myVZw.com"
'Verizon
ElseIf (msgCarrier = "Voicestream") Then
maxmsgLength = 140
msgLength = "voicestream.net"
' Default Length
Else
maxLength = 140
'put a valid default domain
End If
```

Build and send the message

The final task is to build the message and send it. The length of the message is compared to the length that the carrier will accommodate.

```
  'Build the message

  msgLength = len(msgTxt)

  If (msgLength > maxmsgLength) Then
  Response.Write "Please make sure that your message is not longer than" & maxmsgLength
& "characters."
  Else
  sendmail.From = msgFrom
  sendmail.To = msgTo & "@" & msgDomain
  sendmail.Body = msgTxt

  'Send The Short Text Message

  sendmail.Importance = 2  'Priority -> 0=low 1=normal 2=high
  sendmail.Send

  set sendmail = nothing

  Response.Write "Message has been sent to" & "<br/>" & msgTo
  End If
```

The entire short text message application

Copy and save the following code as sendSMS.asp.

```
  <%@ Language = "VBscript"%>

  <%
  dim msgFrom
  dim msgTo
  dim msgCarrier
  dim msgTxt

  'Retrieve message values from the form

  msgFrom = Request("from")
  msgTo  = Request("to")
  msgCarrier = Request("carrier")
  msgTxt  = Request("text")

  'Create message instance using cdonts object

  Set sendmail = Server.CreateObject("CDONTS.NewMail")

  'Set message lengths associated with each carrier
  ' AT&T
  If (msgCarrier = "ATT") Then
  maxmsgLength = 140
  msgDomain = "mobile.att.net"
  ' Nextel
  ElseIf (msgCarrier = "Nextel") Then
  maxmsgLength = 280
  msgDomain = "messaging.nextel.com"
  ' Sprint PCS
  ElseIf (msgCarrier = "Sprint") Then
  maxmsgLength = 100
  msgDomain = "messaging.sprintpcs.com"
  'Verizon
  ElseIf (msgCarrier = "Verizon") Then
  maxmsgLength = 120
  msgDomain = "msg.myVZw.com"
```

```
'Verizon
ElseIf (msgCarrier = "Voicestream") Then
maxmsgLength = 140
msgLength = "voicestream.net"
' Default Length
Else
maxLength = 140
'put a valid default domain
End If

'Build the message

msgLength = len(msgTxt)

If (msgLength > maxmsgLength) Then
Response.Write "Please make sure that your message is not longer than" & maxmsgLength & "characters."
Else
sendmail.From = msgFrom
sendmail.To = msgTo & "@" & msgDomain
sendmail.Body = msgTxt

'Send The Short Text Message

sendmail.Importance = 2  'Priority -> 0=low 1=normal 2=high
sendmail.Send

set sendmail = nothing

Response.Write "Message has been sent to" & "<br/>" & msgTo
End If
%>
```

Testing the application

You can test the sendSMS.asp message implementation with the following HTML code. The form takes as an input To, From, Carrier name, and the body of the text message. Copy and save the file as sendMail.html.

Note: When testing the application, you might notice that the message does not get to the phone instantaneously. This does not mean that your application is not working. It could simply be due to a delay in transmission through the carrier's network.

```
<HTML>
 <BODY>
 <FORM action="sendSMS.asp" method="post">
 <TABLE align="center">
  <TR>
   <TD width="50%">
    To:<BR><INPUT name="to" size="25">
   </TD>
   <TD width="50%">
    From:<BR><INPUT name="from" size="25">
   </TD>
  </TR>
  <TR>
<TD>Carrier:<BR>
<SELECT  name="carrier">
<OPTION value="ATT">AT&T
<OPTION value="Sprint">Sprint PCS
<OPTION value="Nextel">Nextel
<OPTION value="Verizon">Verizon
<OPTION value="Voicestream">Voicestream
</SELECT>
</TD>
</TR>
  <TR>
   <TD colspan="2">
```

```
      <P>Message:<BR><TEXTAREA name="text" rows=4 cols=35></TEXTAREA></P>
     </TD>
    </TR>
   </TABLE>
   <CENTER><INPUT type="submit" value=" Send Message "></CENTER>
  </FORM>
  </BODY>
  </HTML>
```

Short text message via SMTP -- An example using JSP

The example uses Sun's JavaMail API for e-mail handling. JavaMail is not included in the standard Java Development Kit. JavaMail and JavaBeans Activation Framework (JAF) need to be downloaded and added to the CLASSPATH for Windows 2000, or as appropriate for other operating systems. (See the Resources section for more information on JavaMail and JAF.) In order to send a message using JavaMail, four main components are required: Properties, Session, Transport, and Message.

Outline of JSP page

The outline of the application is as follows:

```
<%@ page import=" javax.mail.*, javax.mail.internet.*,
javax.activation.*,java.util.*" %> <HTML> <BODY> <% try\{
'Variable declaration . . 'Create instance of message . .
'Retrieve message information and build the message to be sent .
. 'Send the message . . %>
```

Properties class

The Properties class is used to create a session object that holds the SMTP server name. Replace "smtp.hostname.com" with a known SMTP server. `Properties mailserver = new Properties (); mailserver.put("mail.smtp.host", "smtp.hostname.com");`

Session class

The Session class is used to create a mail session that retrieves the SMTP server name for the mail transaction. `Session sendMailSession; sendMailSession = Session.getInstance(mailserver,null);`

Transport class

The Transport class is used to send the message. You will have to provide the protocol being used, i.e. POP3, SMTP, or IMAP. `Transport transmit; transmit = sendMailSession.getTransport("smtp")`

Message class

The Message object holds all the information of the actual message being sent out, i.e. TO, FROM, TEXT. The Message object is created as a MimeMessage. `Message newMessage = new MimeMessage(sendMailSession);`

Create message instance

The following lines of code are used to create a mail session and assign an SMTP server. Make sure you use a valid SMTP server name. When you run this example you will get an exception error if you don't use a valid SMTP server. `sendMailSession = Session.getInstance(mailserver, null); mailserver.put("mail.smtp.host", "smtp.hostname.com");`

Building, creating, and sending the message

The past few panels discussed the main objects used for building, creating, and sending the short text message. The 'get' and 'set' properties of the message object are used to build the message. The request.getParameter() is used to retrieve information from the HTML form.

```
// retrieve the information submitted through the sendmail.html page

Message newMessage = new MimeMessage(sendMailSession);
newMessage.setFrom(new InternetAddress(request.getParameter("from")));
newMessage.setRecipient(Message.RecipientType.TO, new InternetAddress(request.getParameter("to")));
newMessage.setSentDate(new Date());
newMessage.setText(request.getParameter("text"));

// send the short text message

transmit = sendMailSession.getTransport("smtp");
transmit.send(newMessage);
```

The entire short text message application

Below are all the pieces assembled to create the JSP page for sending a short text message. Copy and save the following code as sendSMS.jsp.

```
<%@ page
  import=" javax.mail.*, javax.mail.internet.*, javax.activation.*,java.util.*"
  %>
<HTML>
<BODY>
<%

 try\{
   Properties mailserver = new Properties();
   Session sendMailSession;
   Transport transmit;

  sendMailSession = Session.getInstance(mailserver, null);
  mailserver.put("mail.smtp.host", "smtp.hostname.com");

  // retrieve the information submitted through the sendmail.html page

  Message newMessage = new MimeMessage(sendMailSession);
  newMessage.setFrom(new InternetAddress(request.getParameter("from")));
  newMessage.setRecipient(Message.RecipientType.TO, new InternetAddress(request.getParameter("to")));
  newMessage.setSentDate(new Date());
  newMessage.setText(request.getParameter("text"));

  // send the short text message

  transmit = sendMailSession.getTransport("smtp");
  transmit.send(newMessage);
   %>
<P>Your mail has been sent.</P>
>%
  \}
 catch(MessagingException m)
  \{
  out.println(m.toString());
  \}
%>
</BODY>
</HTML>
```

Testing the application

Use the following form to post the short text message's information to the sendSMS.jsp page. The form takes as an input To, From, and the body of the text message. Copy and save the file as sendMail.html.

Don't forget to add the carrier's domain name after the phone number. For example, if you are sending a message to an AT&T phone, the e-mail address will be xxxxxxxxxx@mobile.att.net (no spaces).

```
<HTML>
 <BODY>
 <FORM action="sendSMS.jsp" method="post">
 <TABLE align="center">
  <TR>
   <TD width="50%">
    To:<BR><INPUT name="to" size="25">
   </TD>
   <TD width="50%">
    From:<BR><INPUT name="from" size="25">
   </TD>
  </TR>
  <TR>
   <TD colspan="2">
    <P>Message:<BR><TEXTAREA name="text" rows=4 cols=35></TEXTAREA></P>
   </TD>
  </TR>
 </TABLE>
 <CENTER><INPUT type="submit" value=" Send Message "></CENTER>
</FORM>
</BODY>
</HTML>
```

SMS via a mobile terminal

Short messages can be sent from a PC program to a mobile phone. You are required to install the mobile phone as a modem to the PC's operating system. The phone is connected to the PC with a cable. The following is an illustration of how you would go about sending text mode SMS from a terminal program like Hyper Terminal:

```
'Send an 'AT' command to the phone AT (press "enter") 'Set the
SMS sending mode to text mode. 0 is for PDU mode. AT+CMGF=1
(press "enter") 'The message is sent to the phone number.
(ctrl-z) indicates the end of the message.
AT+CMGS="+18005551212" (press "enter") ("input some text")
(ctrl-z) AT OK AT+CMGF=1 OK AT+CMGS="+18005551212",129 >Hello
+CMGS:3 OK
```

SMS tools and gateways

Here are a few SMS tools. See the Resources section for more information:

* G@te, by Magic4.com
* SMPP Developer's Tool Kit, by Logica
* SMS-IT, by EMD Group

Here are some SMS gateways:

* SMS Gateway, by Winsms
* Empowered SMS Gateway, by Empower Interactive Group Ltd.

Section 6. Comparison to other network services and protocols

SMS and network protocols

In the next few sections SMS will be compared with a paging service and other network protocols, like Wireless Application Protocol(WAP) and General Packet Radio Services(GPRS).

SMS and paging

	SMS	**Paging**
Service design	A message originates either from a mobile phone, e-mail, Web site, or other short messaging entity (SME) and is sent to a mobile device via the short messaging service center (SMSC).	A message orig e-mail, or Web telephone netw (PSDN) to a pa the message th
Message Length	160 Characters	Varies by pagin
Protocol	SMS specific (e.g., SMPP)	TAP, TME for a
Delivery Confirmation	SMS features confirmation of message delivery	Paging does no
Interactivity	One and two-way SMS available	One and two-w
Terminals	Web, e-mail, mobile phone	Web, e-mail, m
Message Type	Alphanumeric	Alphanumeric

SMS and WAP

	SMS	**WAP**
Length	160 Characters	Compiled page
Browser	Mobile device does not need a browser to receive and send SMS messages	Mobile device r
Service Type	SMS is a bearer service. SMS is a mechanism of delivering short messages	WAP provides
Information Delivery	Can receive an SMS message while on a voice call	Cannot use WA time
Service Delivery	Uses store and forward mechanism	Data sent and c
Technology Type	Applications can be implemented in Visual Basic, Java, Perl, or PHP	Applications im

SMS and GPRS

	SMS	**GPRS**

Message Length	Layer 3 message length extends to 140 octets, or 160 characters	Layer 3 messa
Service Type	SMS is a bearer service. SMS is a mechanism of delivering short messages.	GPRS is a pac
Service Delivery	SMS uses a store and forward mechanism. Therefore, the mobile phone is not required to be active and within range to deliver the message. Once the phone is active and within range the message is delivered.	GPRS provides connected to th
Service Offering	Due to length of message, SMS would be the bearer of choice to provide services like notifying users about certain events, or prompting a service to users (e.g. account information, stocks, etc.) via a short text message.	GPRS would b WAP-enabled
Availability	SMS has been available since 1991 and is experiencing dramatic growth throughout the world.	GPRS deploym available since

Section 7. Conclusion

Advantages of SMS

* The store and forward mechanism is very useful when a recipient is not available.
* SMS provides a reliable and low-cost communication method for short message delivery.
* Simultaneous message delivery to multiple subscribers.
* Message delivery to multiple subscribers at a time.
* Can easily be integrated with Internet-based or other data applications.
* Provides another source of revenue and creation of new services for the service provider.

Limitations of SMS

* The message length is limited to 160 characters; this is ideal only for simple text messages.
* SMS does not support audio or graphics.
* The store and forward mechanism of SMS, though very useful, does not make it suitable for WAP applications.
* Slow data rate and latency. The signaling channel used by SMS is used for other purposes, which tends to slow the message transmission data rate.
* The SMS protocol data unit, as defined in the GSM 03.40 standard, is not flexible. Header fields, including Data Coding Scheme and Origination Address, are fixed. This can sometimes constrain application development. 3G specifications will include a Tag Length Variable structure to address the SMS message structure's inflexibility.

Future of SMS

At this time, SMS delivers only alphanumeric messages. In the future, SMS will evolve to provide Enhanced Messaging Service (EMS) and Multimedia Messaging Service (MMS). EMS should be made available soon, and will provide much richer content, including melodies, pictures, sounds, and animation. MMS will be the next evolution after EMS and will provide sounds, images, and video. MMS will probably hit the market toward the end of 2002.

Additional resources

For additional information, resources, and SMS specifications, refer to the following sites:

* For more information on SS7 links, see *www.webproforum.com/ss7/* .
* For more information on JavaMail go to *http://java.sun.com/products/javamail/index.html* , and for JAF go to *http://java.sun.com/products/javabeans/glasgow/jaf.html* .
* *www.etsi.org*
* *www.gsmworld.com*
* *forum.nokia.com*
* G@te, by Magic4.com (*www.magic4.com*)
* SMPP Developer's Tool Kit, by Logica (*www.logica.com*)
* SMS-IT, by EMD Group (*www.sms-it.com*)
* SMS Gateway, by Winsms (*www.winsms.com*)
* Empowered SMS Gateway, by Empower Interactive Group Ltd (*www.eigroup.com*)
* *Case Study: IBM Networking Software and SMS*

Section 8. Feedback

Feedback

Please let us know whether this tutorial was helpful to you and how we could make it better. Feel free to suggest improvements or other tutorial topics you'd like to see covered. Thanks!

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The Toot-O-Matic tool is a short Java program that uses XSLT stylesheets to convert the XML source into a number of HTML pages, a zip file, JPEG heading graphics, and PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML.