ActiveXperts Mobile Messaging Toolkit

Improve your software with SMS, Social Media and E-mail capabilities.

The Mobile Messaging Toolkit makes integrating SMS, Social Media and / or E-mail into your application a trivial matter. With social media integration it is now possible to reach, alert or find anyone who uses either Twitter, Linked-In or Facebook.

Mobile Messaging Toolkit Overview

The Mobile Messaging Toolkit is an ActiveX/COM component that is accessible through almost all platforms that are available for Microsoft Windows.

It's easy to use the Mobile Messaging Toolkit through the large amount of working examples that are included. The product ships with sample source code for various development platforms, including:

- Visual C# .NET and Visual Basic .NET
- ASP .NET (C#/VB) and ASP 2.x
- Visual C++
- VBScript and VBA (Visual Basic for Applications)
- Delphi
- HTML / Javascript
- PHP
- ColdFusion

The Mobile Messaging Toolkit design has a number of important features:

- Troubleshooting. Protocol level logging is offered on all messaging options in the toolkit
- Portability. Since all functionality is included in a single .DLL file the deployment of the toolkit in your own product is a breeze
- Thread-safety. All protocols are completely thread-safe and can be used in multi-threaded environments
- ActiveX/COM. Nearly all programming and scripting environments on Windows have built-in means of accessing ActiveX/COM components

SMS and Pager Features

Today nearly everyone can be reached through their mobile and nearly all mobile devices have support for SMS. SMS and pager support is an important part of the Mobile Messaging Toolkit.

This shows through the large number of SMS protocols that are supported and the maturity level of the SMS protocol implementations as well as the high-level API.

The following SMS and Pager protocols are supported: GSM, SMPP (client and server), HTTP, Dialup (TAP/XIO), SNPP.

Each of these protocols is easily accessible. This is an example of sending an SMS message through the GSM protocol:

Send an SMS using a GSM modem

```
Set objGsm = CreateObject( "AxMmToolkit.Gsm" )              ' Create the GSM protocol object
Set objMessage = CreateObject( "AxMmToolkit.SmsMessage" )   ' Create the SMS message object

sDevName = objGsm.FindFirstDevice                           ' Find the first connected TAPI device
objGsm.Open sDevName                                        ' Open this device
WScript.Echo "Open, result: " & objGsm.LastError

objMessage.ToAddress = "+3611223344"                        ' Compose an SMS message, set address
objMessage.Body = "This is a short text message"            ' Compose an SMS message, set body

objGsm.SendSms objMessage                                   ' Send SMS message to the GSM modem
WScript.Echo "SendSms, result: " & objGsm.LastError

objGsm.Close                                                ' Close the connection
```

The following is a list of commonly used SMS features:

- Unicode and Binary SMS support for GSM, SMPP and HTTP
- Delivery reports are supported for both GSM and SMPP
- SMPP Server. Powerful SMPP server functionality to build your own SMSC
- Application ports as supported for GSM and SMPP. Set the port in the address. E.g.: '+316223344:1000'
- Multipart messages are automatically split up and assembled for GSM and SMPP
- Both 8bit and 16bit UDH multipart modes supported for GSM
- 8 / 16bit UDH, SAR TLV and Payload TLV multipart modes supported for SMPP
- SMPP versions 5.0, 3.4 and 3.3 supported
- TLV's are supported in SMPP. TLV's are easily accessible for reading and setting
- Data_sm and submit_sm packets are supported for sending and receiving messages
- WAPPush and vCard templates. To easily format these binary SMS types

An example of sending an SMS message through SMPP:

Send an SMS using SMPP

```
Set objSmpp = CreateObject( "AxMmToolkit.Smpp" )            ' Create the SMPP protocol object
Set objConst = CreateObject( "AxMmToolkit.SmsConstants" )   ' Create the SMS constants object
Set objMessage = CreateObject( "AxMmToolkit.SmsMessage" )   ' Create the SMS message object

objSmpp.Connect "smpp.activexperts-labs.com", 2775, 2000    ' Connect to the SMSC
WScript.Echo "Open, result: " & objSmpp.LastError

objSmpp.Bind objConst.SMPP_BIND_TRANSCEIVER, _              ' Login to the SMSC
  "CE658B84", "FAC1982E", "", objConst.SMPP_VERSION_34, 0, 0, "", 2000
WScript.Echo "Bind, result: " & objSmpp.LastError

objMessage.ToAddress = "+3611223344"                        ' Compose an SMS message, set address
objMessage.Body = "This is a short text message"            ' Compose an SMS message, set body

objSmpp.SubmitSms objMessage                                ' Submit the message to the SMSC
WScript.Echo "SubmitSms, result: " & objSmpp.LastError

objSmpp.WaitForSmsUpdate( 5000 )                            ' Wait up to 5000 seconds for an update
Set objMessage = objsmpp.FetchSmsUpdate                     ' Fetch the updated message status
WScript.Echo "Status: " & objMessage.SmppCommandStatus      ' See if the SMSC has accepted the SMS
WScript.Echo "Reference: " & objMessage.Reference           ' Display the SMSC message reference

objSmpp.Unbind                                              ' Log out
objSmpp.Disconnect                                          ' Disconnection from the server
```

Social Media features

Social Media is rapidly gaining popularity as a primary messaging platform for people in areas where internet coverage is ubiquitous.

There are currently three important platforms that compose the bulk of the social media spectrum and its participants: Facebook, Twitter and Linked in. All three are fully supported by the Mobile Messaging Toolkit.

These platforms are very important as a means of reaching or alerting people as well as finding out more about potential customers.

Here's an example of authenticating and sending a status update using Twitter in the Mobile Messaging Toolkit:

Authenticate on Twitter

```
Set objTwitter = CreateObject( "AxMmToolkit.Twitter" )     ' Create the Twitter object

objTwitter.LoadConsumerKey "Consumer.key"                  ' Load your application key
WScript.Echo "LoadConsumerKey, result: " & objTwitter.LastError

strUrl = objTwitter.RequestAuthorization                   ' Request an authorization URL
WScript.Echo "RequestAuthorization, result: " & objTwitter.LastError

Set objIE = CreateObject( "InternetExplorer.Application" ) ' Create the Internet explorer object
objIE.Navigate strUrl: objIE.Visible = 1                   ' Use IE to have the user log in
Do While objIE.Busy: Loop                                  ' Wait until the log in page is loaded

strVerify = inputbox( "Enter verifier", "verifier", "" )   ' Obtain the verification code
objTwitter.RequestAccessToken strVerify                    ' Trade verifier for an access token
WScript.Echo "RequestAccessToken, result: " & objTwitter.LastError

objTwitter.Tweet "Tweet about our success !"               ' Send out a Tweet
WScript.Echo "Tweet, result: " & objTwitter.LastError
```

The following is a list of commonly used social media features:

- OAuth 1.0a as well as OAuth 2.0 are supported for authenticating users
- Client and server authentication supported for both OAuth 1.0a and OAuth 2.0. It's as easy to write a client application as it is to write a web application
- XML as well as Json results for all media. Including Facebook. Only one parser needed to use any API results
- API Call functions for every protocol. This enables the use of any and all documented calls for Twitter, Linked-In and Facebook
- Unicode

This is an example of listing all Facebook friends in the current user profile:

List friends on facebook

```
Set objFacebook = WScript.CreateObject( "AxMmToolkit.Facebook" ) ' Create Facebook protocol object

objFacebook.AccessToken = "AABFHg1TOeSgBAC8EEZA...c27PppBCHiVp4MUUAZDZD"

Set objNode = objFacebook.FindFirstFriend                  ' Retrieve a list of friends
While objFaceBook.LastError = 0
  WScript.Echo "Id: " & objNode.Id                         ' Display the friend Id
  WScript.Echo "Name: " & objNode.Name                     ' Display the friend Name
  Set objNode = objFacebook.FindNextFriend                 ' Iterate to the next friend
WEnd
```

E-mail features

E-mail is an integral part of today's society. It has all but replaced the traditional means of sending messages on paper media. We have come to expect being able to share anything with anyone in just a moment's notice.

This is why full featured and easy to use e-mail support is an important part of the Mobile Messaging Toolkit. The Mobile Messaging Toolkit supports SMTP for sending and POP3 for receiving. E-mail messages in the Mobile Messaging Toolkit expose most Mime related features.

The following is a list of commonly used features social media features:

- TLS/SSL support on both SMTP and POP3
- Gmail and Windows Live Mail are supported as well as most other popular cloud based e-mail services
- CRAM-MD5, Login and Plain authentication algorithms are supported for SMTP
- APOP3 and Plain text authentication supported for POP3
- Loading and Saving from- and to MIME files
- HTML and Plain-text bodies as well as a combination of HTML with a plain-text alternative are supported
- Attachments

Here's an example of sending out an e-mail using SMTP:

Send an E-mail using SMTP

```
Set objSmtp = CreateObject( "AxMmToolkit.Smtp" )           ' Create the SMTP Protocol object
Set objMail = CreateObject( "AxMmToolkit.EmailMessage" )   ' Create the e-mail message object

objSmtp.SetSecure                                          ' Enabled SSL/TLS connection
objSmtp.Connect "smtp.gmail.com", "me@gmail.com", "password" ' Connect to the SMTP server
WScript.Echo "Connect, result: " & objTwitter.LastError

objMail.FromAddress = "me@gmail.com"                       ' Compose an e-mail
objMail.FromName = "My Name"                               ' Set the from name
objMail.AddTo "someone@example.com", "Someone"             ' Set the to address
objMail.Subject = "Hi!"                                    ' Set the subject text
objMail.BodyHtml = "<html><body><h1>Hi!</h1></body></html>" ' Set the body text
objMail.BodyPlainText = "Hi!"                              ' Set a plain text alternative

objSmtp.Send objMail                                       ' Send the e-mail out
WScript.Echo "Send, result: " & objTwitter.LastError

objSmtp.Disconnect                                         ' Disconnect from the server
```

Here's an example of opening a POP3 inbox:

Receive E-mail using POP3

```
Set objPop3 = CreateObject( "AxMmToolkit.Pop3" )           ' Create the POP3 protocol object
Set objMail = CreateObject( "AxMmToolkit.EmailMessage" )   ' Create the e-mail message object

objPop3.SetSecure                                          ' Enable SSL / TLS and set port
objPop3.Connect "pop.gmail.com", "me@gmail.com", "password"  ' Connect to the POP3 server
WScript.Echo "Connect, result: " & objPop3.LastError

For i = 1 to objPop3.CountMessages()                       ' Iterate over all messages
  Set objMail = objPop3.GetEmailMessage(i)                 ' Retrieve the message
  WScript.Echo "GetEmailMessage, result: " & objPop3.LastError

  WScript.Echo "----"
  WScript.Echo objMail.FromAddress                         ' Display the from address
  WScript.Echo objMail.Subject                             ' Display the subject
  WScript.Echo objMail.BodyPlainText                       ' Display the plain-text part

  For j = 1 to objMail.CountAttachments()                  ' Iterate over all attachments
    strName = objMail.GetAttachmentName(j)                 ' Get the (file)name
    WScript.Echo "  Attachment found: " & strName          ' Display the name
  Next
Next

objPop3.Disconnect                                         ' Disconnect from the server
```

Visit **www.activexperts.com/mobile-messaging-component** for more information.

Download your evaluation version at www.activexperts.com/download
Requirements: Windows 2008 (R2) / 2003 / 8 / 7 / Vista / XP, 100MB of free disk space