ActiveXperts Software Components

Advanced networking and communications capabilities

Easily integrate advanced networking and communications features into your applications. ActiveXperts software components are powerful, versatile and currently in use in numerous applications across the globe.

Software components overview

The core functionality and all protocol knowledge for ActiveXperts desktop applications are implemented our software components.

There are three components:
- Mobile Messaging Toolkit. SMS, Social Media and E-mail
- ActiveSocket. A rich collection of network communications protocols.
- ActiveComport. Serial port and TAPI communications

All components are ActiveX/COM components. This means they can be accessed in nearly all platforms that are available for Microsoft Windows.

Integrating ActiveXperts components in your own application is made easy by the large number of samples that are included. Every component includes working examples for all of these platforms and more:
- Visual C# .NET
- Visual Basic .NET
- ASP .NET (C# / VB)
- ASP 2.x
- Visual C++
- VBScript
- PowerShell
- VBA (Visual Basic for Applications)
- Borland Delphi
- HTML / JavaScript

In addition, each component supports the following features:
- Troubleshooting. Detailed logging is included in all protocols in all of our toolkits
- Portability. Every component has all of its functionality in a single .DLL file. This makes deploying one of our components as part of your own application a breeze
- Thread-safety. All protocols in our components are completely thread-safe and can be used in multi-threaded environments
- ActiveX/COM. Almost all programming and scripting environments on Windows have built-in means of accessing ActiveX/COM components

Mobile Messaging Toolkit

This component has its focus on mobile messaging, that is, SMS, social media and e-mail.

The mobile messaging toolkit includes support for the following messaging options:
- SMS. Messages can be sent through GSM, HTTP, SMPP (Client and Server) and Dialup (TAP / XIO) connections.
- Pager. Use SNPP to send pager messages
- Social media. This includes support for Facebook, Twitter and Linked-In
- E-mail. Send through SMTP or receive through POP3

Each of these protocols is easily accessible. This is an example of sending an SMS message through the GSM protocol using a pre-configured TAPI device:

```
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
```

An example of authenticating and sending a status update through Twitter:

```
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
```

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

ActiveSocket

This component implements a large number of IP-based protocols. Its main focus is monitoring and communicating with existing servers in a network.

An overview of the implemented protocols:

- DNS. Query servers running a domain name service
- FTP. Read and write files on a remote server
- HTTP. Quick and easy HTTP, includes connection pooling
- ICMP. Ping any server or workstation from your application
- IPtoCountry: Translate an IP address to a location
- MSN. Connect to Microsoft Live Messaging Service
- NTP. Query time services for the current time
- RADIUS. Centralized authentication and authorization
- RSH. Run commands and scripts on a remote Unix machine
- SCP. Secure file transfer protocol based on SSH
- SFTP. A more advanced secure file transfer protocol
- SNMP. V1 and V2 of the Simple Network Management Protocol. Supports SNMP Traps and MIB browsing
- SSH. Modern, secure alternative to RSH
- TCP. Write your own TCP-based client/server application
- TFTP. Get or put single files for a remote TFTP host
- TraceRoute. Find the route to a specific host
- UDP. Create UDP based client/server applications
- Wake-On-LAN. Remote power-up machines on your LAN

SNMP is a widely used protocol to monitor hardware and software. Almost any device within reach of a network administrator supports SNMP to monitor or manager its properties.

ActiveSocket makes SNMP easy by offering, among its other protocols, a high-level API for getting and setting SNMP objects as well as sending and receiving SNMP Traps.

Here's an example that opens an MIB file and walks through all of its objects to print every value:

```
SNMP walker

Set objSnmp = CreateObject ( "ActiveXperts.SnmpManager" )   ' Create the SNMP protocol object
Set objConst = CreateObject ( "ActiveXperts.ASConstants" )  ' Create the constants object

objSnmp.Initialize                                          ' Initialize SNMP library
WScript.Echo "Initialize, result: " & objSnmp.LastError

objSnmp.LoadMibFile( "test.mib" )                           ' Load the MIB file
WScript.Echo "LoadMibFile, result: " & objSnmp.LastError

objSnmp.Open "localhost", "public"                          ' Initialize SNMP connection
WScript.Echo "Open, result: " & objSnmp.LastError

Set objSnmpObject = objSnmp.Get("system.sysName.0")         ' Get the first SNMP object
While( objSnmp.LastError = 0 )
  WScript.Echo "Name :" & objSnmpObject.OIDName             ' Display the name
  WScript.Echo "Name :" & objSnmpObject.Value               ' Display the value
  Set objSnmpObject = objSnmp.GetNext                       ' Get the next SNMP object
WEnd

objSnmp.Close                                               ' Close the connection
objSnmp.Shutdown                                            ' Shutdown the library
```

HTTP is another protocol that's widely used today. From serving HTML pages to the end-user to being used as the primary carrier for SOAP and REST based web service API's. The ActiveSocket HTTP protocol implementation is particularly strong in its simplicity and high performance.

Smart connection pooling makes it a strong bet when performance is important. Whenever possible the HTTP component in ActiveSocket tries to re-use active HTTP sessions.

Here's an example that demonstrates how connection pooling is implemented:

```
Demonstrate HTTP connection pooling

Set objHttp = CreateObject( "ActiveXperts.HttpEx" )         ' Create the HTTP protocol object

objHttp.LogFile = "ConnectionPoolingText.Log"               ' Keep a log file

objHttp.ConnectionPoolSize = 5                              ' Set connection pool size
objHttp.ConnectionExpireTimeout = 5000                      ' Set connection pool expire timeout

WScript.Echo objHttp.Get( "www.google.com" )                ' Get the Google page
objHttp.Sleep 1000                                          ' Wait 1 second ...
WScript.Echo objHttp.Get( "www.activexperts.com" )          ' Get the ActiveXperts page
objHttp.Sleep 1000                                          ' Wait 1 second ...
WScript.Echo objHttp.Get( "www.google.com" )                ' Re-use the connection for Google
objHttp.Sleep 1000                                          ' Wait 1 second ...
WScript.Echo objHttp.Get( "www.slashdot.com" )              ' Get the Slashdot page
objHttp.Sleep 1000                                          ' Wait 1 second ...
WScript.Echo objHttp.Get( "www.google.com" )                ' Re-use the connection for Google
objHttp.Sleep 1000                                          ' Wait 1 second ...
WScript.Echo objHttp.Get( "www.tweakers.net" )              ' Get the Tweakers page
objHttp.Sleep 1000                                          ' Wait 1 second ...
WScript.Echo objHttp.Get( "www.google.com" )                ' Re-use the connection for Google
objHttp.Sleep 1000                                          ' Wait 1 second ...
WScript.Echo objHttp.Get( "www.activexperts.com" )          ' Get the ActiveXperts page (again)
```

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

ActiveComport

Connect to any RS232 based device with just a few lines of code.

The following are commonly used features:

- TAPI support. Open a TAPI device just like a COM device
- Flow control. Use either hardware or software flow control
- Text and binary. Send and receive any kind of data

This is an example which opens a comport to reset a modem:

```
Open a modem using ActiveComport

Set objCom = CreateObject( "ActiveXperts.Comport" )              ' Create a new Comport instance

objCom.Device              = "COM1"                              ' Use a COM port directly
objCom.BaudRate            = 56000                               ' Set baudrate
objCom.HardwareFlowControl = objCom.asFLOWCONTROL_ENABLE         ' Set Hardware Flow Control
objCom.SoftwareFlowControl = objCom.asFLOWCONTROL_ENABLE         ' Set Software Flow Control

objCom.Open                                                      ' Open the port
WScript.Echo "Open, result: " & objCom.LastError

objCom.WriteString( "AT&F" )                                     ' Write command to reset modem
str = objCom.ReadString
WScript.Echo "Received: [" & str & "]"                           ' Read the response

objCom.Close                                                     ' Close the port
```

Visit **www.activexperts.com/serial-port-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