

# ActiveXperts Serial Port Component Manual

© [ActiveXperts Software B.V.](http://www.activexperts.com) [contact@activexperts.com](mailto:contact@activexperts.com)

## 1. Introduction

### 1.1. Introduction

Adding serial communications capabilities to an application is never a simple matter. It requires specialized knowledge that might be outside an individual programmer's expertise. For years, VBScript, Visual Basic and Visual C++ developers have relied upon the power, flexibility and reliability of the ActiveXperts Serial Port Component serial communications control from ActiveXperts Software. And today, also .NET developers use this control.

ActiveXperts Serial Port Component is a COM component, that provides an easy-to-use scripting interface for serial, asynchronous communications through a serial port. ActiveXperts Serial Port Component can control modems, ISDN modems, USB serial devices and other devices and machines that have a serial interface.

Use ActiveXperts Serial Port Component for different purposes:

- To control manufacturing machines via the serial port;
- To configure network devices (like print-servers, routers) via the serial port;
- To control a modem, connected to the serial/USB port or Bluetooth;
- To transfer files through a null modem cable;
- Any other scenario where serial communications is involved.

ActiveXperts Serial Port Component features the following:

- Direct COM ports supported (like 'COM2');
- Windows Telephony Devices supported (like 'Standard 9600 bps Modem');
- Support for RS-232, RS422 and RS485;
- Up to 256 ports opened simultaneously;
- Thread-safe to allow the toolkit in multi-threading environments (multi-threading samples included);
- Support for Hayes compatible modems, connected via a serial port, USB or Bluetooth;
- Support for GSM/GPRS modems (serial port, USB or Bluetooth);
- Support for Virtual COM ports (i.e. COM ports redirected through the network);
- Hardware flow control (RTS/CTS, DTR/DSR);
- Software flowcontrol (XON/XOFF);
- Support for any baudrate;
- Ability to set baudrates, parity, stopbits;
- Full buffered data transfer;
- Text and Binary data transfer;
- Advanced logging.

ActiveXperts Serial Port Component includes [samples](#) for many development tools, including:

- Visual Basic .NET - Windows .NET based application;
- Visual C# .NET - Windows .NET based applications;
- Visual Basic 6.x or higher - Windows based applications;
- Visual C++ 6.x or higher - Windows based applications;
- ASP .NET - Web site based on Active Server Pages and the .NET Framework;
- ASP 2.x - Web site based on Active Server Pages (server-side scripting);
- PHP - Embedded HTML scripting;

- PowerShell - Windows based scripts;
- VBScript - Windows based scripts;
- Java/Javascript - Java based scripts;
- HTML - Client scripts within HTML pages;
- Delphi - Windows based applications;
- Any other development platform that supports ActiveX/COM components.

## 1.2. Architecture

ActiveXperts Serial Port Component is built on top of the Microsoft serial device drivers. It just uses these drivers. It neither replaces them, nor does it install any additional serial device drivers.

The core of ActiveXperts Serial Port Component is an ActiveX/COM component that comes in a 32-bit and a 64-bit version:

- AxSerial32.dll - the 'ActiveXperts Serial Port Component COM Component' for 32-bit platforms;
- AxSerial64.dll - the 'ActiveXperts Serial Port Component COM Component' for 64-bit platforms;

ActiveXperts Serial Port Component can be distributed easily to many PC's. Once you have purchased the licenses, you copy the AxSerial32.dll (or AxSerial64.dll) to the PCs and register the DLL on that PC.

## 2. System Requirements

### 2.1. Operating System

ActiveXperts Serial Port Component can be used on any of the following operating systems:

- Windows Server 2012 (64-bit)
- Windows Server 2008R2 (64-bit)
- Windows Server 2008 (32-bit and 64-bit)
- Windows Server 2003 (32-bit and 64-bit)
- Windows 8 (32-bit and 64-bit)
- Windows 7 (32-bit and 64-bit)
- Windows Vista (32-bit and 64-bit)
- Windows XP (32-bit and 64-bit)

### 2.2. .NET Framework

To use ActiveXperts Serial Port Component in an ASP .NET, Visual Basic .NET or Visual C#. NET environment, the .NET Framework 2.0 or higher must be installed on the system. The .NET Framework is part of Windows 2003 server platforms and higher, and on Windows Vista workstation platforms and higher. For other Windows platforms, it's available as a separate installation. Please visit the [Technology Information for the .NET Framework](#) page to download the .NET Framework.

### 2.3. Internet Information Server

Internet Information Server (IIS) Setup installs the Visual Basic Script and Java Script engines.

To run ASP/ASP.NET pages on Windows Servers, IIS 6.x must be installed. IIS is an optional 'Role' in Windows Server platforms.

To run ASP .NET samples, .NET Framework 2.0 or higher must be installed.

## 3. Installation

### 3.1. Introduction

The ActiveXperts Serial Port Component package consists of 3 modules; any combination of components can be installed:

- The ActiveXperts Serial Port Component COM components - core components 'AxSerial32.dll' (to embed in 32-bit applications) and 'AxSerial64.dll' (to embed in 64-bit applications);
- The ActiveXperts Serial Port Component Help Files - documentation;
- The ActiveXperts Serial Port Component Sample Files - samples for various development platforms.

### 3.2. Installation (Automatic)

To automatically install ActiveXperts Serial Port Component, download [AxSerialSetup.exe](#) and start the installation. The InstallShield wizard will guide you through the rest of the setup.

On 32-bit operating systems, it will automatically register the 32-bit ActiveX DLL 'AxSerial32.dll', to support 32-bit applications.

On 64-bit operating systems, it will automatically register the 64-bit ActiveX DLL 'AxSerial64.dll', to support 64-bit applications; it will also automatically register the 32-bit ActiveX DLL 'AxSerial32.dll', to support 32-bit applications.

### 3.3. Installation (Manual)

To manually install the ActiveXperts Serial Port Component core files, perform the following actions:

- Perform one installation of ActiveXperts Serial Port Component on an available computer, to extract the 'AxSerial32.dll' and 'AxSerial64.dll' core ActiveX files
- To embed ActiveXperts Serial Port Component in 64-bit applications, copy the 'AxSerial64.dll' ActiveX file to the destination computer hard drive, and register the DLL by running the following command from the command-line on the destination computer:  
`REGSVR32 <dest-location>\AxSerial64.dll`
- To embed ActiveXperts Serial Port Component in 32-bit applications, copy the 'AxSerial32.dll' ActiveX file to the destination computer hard drive, and register the DLL by running the following command from the command-line on the destination computer:  
`REGSVR32 <dest-location>\AxSerial32.dll`

## 4. How to use

### 4.1. Introduction

The following code snippets (VBScript) illustrate how to use ActiveXperts Serial Port Component.

## Initialize a modem using a direct COM port

```

Set objComport = CreateObject( "AxSerial.ComPort" )      ' Create new instance

objComport.Device = "COM1"                               ' Use port directly (no Device Driver)
objComport.BaudRate = 56000                             ' Set baudrate (default:9600)
objComport.HardwareFlowControl = asFLOWCONTROL_ENABLE   ' Set Hardware Flow Ctrl (default:On)
objComport.SoftwareFlowControl = asFLOWCONTROL_ENABLE   ' Set Software Flow Ctrl (default:Off)
objComport.Open                                           ' Open the port
Wscript.Echo "Open, result: " & objComport.LastError
If( objComport.LastError <> 0 ) Then
    WScript.Quit
End If

objComport.WriteString( "at&f" )                        ' Write command
str = objComport.ReadString
Wscript.Echo "Received: [" & str & "]"                  ' Read the response

objComport.Close                                          ' Close the port

```

## Initialize a modem using a Windows Telephony Driver

```

Set objComport = CreateObject("AxSerial.ComPort")        ' Create new instance
objComport.Device = "Standard 9600 bps Modem"            ' Use Standard 9600 bps driver
objComport.Open                                           ' Open the port
Wscript.Echo "Open, result: " & objComport.LastError
If( objComport.LastError <> 0 ) Then
    WScript.Quit
End If

objComport.WriteString( "at&f" )                        ' Write command
str = objComport.ReadString
Wscript.Echo "Received: [" & str & "]"                  ' Read the response

objComport.Close                                          ' Close the port

```

## Send an SMS using a GSM Modem connected to the PC; Enable logging

```

Const RECIPIENT = "+31611223344"
Const MESSAGE = "Hello, world!"

Set objComport = CreateObject( "AxSerial.ComPort" )      ' Create new instance

objComport.Device = "Nokia 6680 SmartPhone"              ' Use the Standard 9600 bps driver
objComport.LogFile = "C:\SerialPort.log"                 ' Enable logging

objComport.Open                                           ' Open the port
Wscript.Echo "Open, result: " & objComport.LastError
If( objComport.LastError <> 0 ) Then
    WScript.Quit
End If

WriteStr objComport, "at+cmgs=" & Chr( 34 ) & strNumber & Chr( 34 )
ReadStr objComport
WriteStr objComport, strMessage
strTermCmd = Chr( 26 )                                     ' Terminate: [ctrl]z and then [enter]
WriteStr objComport, strTermCmd
objComport.Sleep 3000                                     ' Takes a while before GSM phone responds
ReadStr objComport                                         ' +CMGS: expected
ReadStr objComport                                         ' OK expected
objComport.Close                                          ' Close the port

' *****
' Sub Routines
' *****
Sub WriteStr( obj, str )
obj.WriteString str
Wscript.Echo "-> " & str
End Sub

```

```

Sub ReadStr( obj )
str = "notempty"
obj.Sleep 200
Do While str <> ""
str = obj.ReadString
If( str <> "" ) Then
WScript.Echo "<- " & str
End If
Loop
End Sub
' *****

```

## 4.2. Visual Basic .NET

First, make sure the ActiveXperts Serial Port component (AxSerial32.dll) is registered on the machine. In case you didn't use the installation program, be sure you used the REGSVR32.EXE program to register to component.

Then, add a reference to the ActiveXperts Serial Port component using the Visual Basic .NET Solution Explorer:

- Start the Solution Provider, go to the project's 'References' container;
- Choose 'Add Reference' from the context menu;
- From the COM components tab, choose the ActiveXperts Serial Port component.

On top of the code, make this declaration:

```
Imports AxSerial
```

and declare and create an object like this:

```

Dim objComport As ComPort = New ComPort           ' Declaration
objComport = New ComPort                          ' Creation

```

After the declaration and creation of the object, you can use the object in your Visual Basic .NET code. Visual Basic samples are part of the product installation.

They can also be found online: <ftp.activexperts-labs.com/samples/serial-port-component>.

## 4.3. Visual C# .NET

First, make sure the ActiveXperts Serial Port component (AxSerial32.dll) is registered on the machine. In case you didn't use the installation program, be sure you used the REGSVR32.EXE program to register to component.

Then, add a reference to the object using the Visual C# Solution Explorer:

- Start the Solution Provider, go to the project's 'References' container;
- Choose 'Add Reference' from the context menu;
- From the COM components tab, choose the ActiveXperts Serial Port component.

On top of your code, make this declaration:

```
using AxSerial;
```

and declare and create an object like this:

```

ComPort objComport;                               // Declaration
objComport = new ComPort();                        // Creation

```

After the declaration and creation of the object, you can use the object in your Visual C# .NET code. Visual C# .NET samples are part of the product installation.

They can also be found online: <ftp.activexperts-labs.com/samples/serial-port-component>.

## 4.4. Visual Basic

ActiveXperts Serial Port Component can be used in Visual Basic 6.x or higher. In Visual Basic, go to the 'Project/References...' menu item and check the box next to ActiveXperts Serial Port Component Type Library. Now, you can declare and create ActiveXperts Serial Port Component objects.

Create a new ActiveXperts Serial Port Component object using the 'CreateObject' method:

```
Dim objComport As AxSerial.Comport      ' Declaration
Set objComport = CreateObject( "AxSerial.ComPort")  ' Creation
```

After the declaration and creation of the object, you can use the object in your Visual Basic code. Visual Basic samples are part of the product installation.

They can also be found online: <ftp.activexperts-labs.com/samples/serial-port-component>.

## 4.5. Visual C++

ActiveXperts Serial Port Component can be used in Visual C++ projects. To do so:

- Import: AxSerial.tlb
- Include: AxSerialConstants.h

Create the various ActiveXperts Serial Port Component object instances like this:

```
#import "AxSerial.tlb"
#include "AxSerialConstants.h"

CoInitialize( NULL );                                // Initialize COM
AxSerial::IComPortPtr oComport = NULL;                // Declaration
oComport.CreateInstance(__uuidof(AxSerial::ComPort)); // Creation
```

After the declaration and creation of the object, you can use the object in your Visual C++ code. Visual C++ samples are part of the product installation.

They can also be found online: <ftp.activexperts-labs.com/samples/serial-port-component>.

## 4.6. ASP 2.x environment

```
<html>
<body>
Version:
<script language=vbscript runat=server>
    Set objComport = CreateObject( "AxSerial.ComPort" )
    Response.Write objComport.Version
</script>
</body>
</html>
```

ASP samples are part of the product installation.

They can also be found online: <ftp.activexperts-labs.com/samples/serial-port-component>.

# 5. Comport object

## 5.1. Properties

## Properties

Property	Type	Read/Write	Description
<a href="#"><u>Version</u></a>	String	Out	Version number of ActiveXperts Serial Port Component
<a href="#"><u>Build</u></a>	String	Out	Build number of ActiveXperts Serial Port Component
<a href="#"><u>LicenseStatus</u></a>	String	Out	License Status
<a href="#"><u>LicenseKey</u></a>	String	In/Out	License Key
<a href="#"><u>LastError</u></a>	Number	In/Out	Result of the last called method
<a href="#"><u>Device</u></a>	String	In/Out	Device name. Either a direct COM port or a Windows Telephony device name
<a href="#"><u>BaudRate</u></a>	Number	In/Out	The baudrate of the communication session; default: 9600 bps
<a href="#"><u>Databits</u></a>	Number	In/Out	The number of databits; default: 8
<a href="#"><u>Stopbits</u></a>	Number	In/Out	The number of stopbits; default: 1
<a href="#"><u>Parity</u></a>	Boolean	In/Out	Parity; default: False
<a href="#"><u>HardwareFlowControl</u></a>	Number	In/Out	Use Hardware Flow Control
<a href="#"><u>DTRFlowControl</u></a>	Boolean	In/Out	Use advanced Hardware Flow Control flag: Data Terminal Ready (default: True)
<a href="#"><u>RTSFlowControl</u></a>	Boolean	In/Out	Use advanced Hardware Flow Control flag: Request To Send; default: True
<a href="#"><u>CTSFlowControl</u></a>	Boolean	In/Out	Use advanced Hardware Flow Control flag: Clear To Send; default: False
<a href="#"><u>DSRFlowControl</u></a>	Boolean	In/Out	Use advanced Hardware Flow Control flag: DSR (Data Set Ready); default: False
<a href="#"><u>SoftwareFlowControl</u></a>	Number	In/Out	Use Software Flow Control
<a href="#"><u>ComTimeout</u></a>	Number	In/Out	Timeout of ReadString, ReadByte and ReadBytes functions, in millisec; default: 1000 msec
<a href="#"><u>IsOpened</u></a>	Boolean	Out	True if port is opened, otherwise False
<a href="#"><u>PreCommandDelay</u></a>	Number	In/Out	Pre command delay, in milliseconds. Only used with the WriteString method
<a href="#"><u>InterCharDelay</u></a>	Number	In/Out	Delay between each character sent, in milliseconds. Only used with the WriteString method
<a href="#"><u>NewLine</u></a>	String	In/Out	The character sequence that forms a newline
<a href="#"><u>LogFile</u></a>	String	In/Out	All serial port commands, as well as data transfer, is logged to this file

### Version property

Version information of ActiveXperts Serial Port Component. This property is read-only; you cannot assign a value to it.

#### Example:

```
Set objComport = CreateObject("AxSerial.ComPort") ' Create new instance
WScript.Echo "Version: " & objComport.Version
```

### Build property

Build information of ActiveXperts Serial Port Component. This property is read-only; you cannot assign a value to it.

#### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
WScript.Echo "Version: " & objComport.Version
WScript.Echo "Build: " & objComport.Build

```

## LicenseStatus property

The status of your license. In case you have not licensed the product, the property holds the trial expiration date. For details, see [Product Activation](#).

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
WScript.Echo "License Status: " & objComport.LicenseStatus
WScript.Echo "License Key: " & objComport.LicenseKey

```

## LicenseKey property

A license key is required to unlock this component after the trial period has expired. Assign the LicenseKey property every time a new instance of this component is created (see code below). Alternatively, the LicenseKey property can be set automatically. This requires the license key to be stored in the registry. For details, see [Product Activation](#).

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.LicenseKey = "XXXXX-XXXXX-XXXXX"           ' Assign your license key
WScript.Echo "LicenseKey: " & objComport.LicenseKey

```

## LastError property

The result of a previous called method. Use it to check the result of your last [method call](#). A zero indicates: success. Any non-zero value means an error.

The [GetErrorDescription](#) method provides the error description of an error code.

For a complete list of error codes, check out the following page: [www.activeexperts.com/support/errorcodes](http://www.activeexperts.com/support/errorcodes).

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
...
objComport.Open                                       ' Open the port
WScript.Echo "LastError: " & objComport.LastError      ' Display the result
...

```

## Device property

### Description:

Device driver to use.

You can either use a Windows telephony device (recommended) or a physical COM port (directly).

Assign one of the following strings to the 'Device' property:

- A valid Windows telephony device name - this must be the literal name as it appears in *Modems* tab of the *Phone and Modems Options* applet in the Control Panel. For instance: "Standard 9600 bps Modem"; Use the [GetDevice](#) method to retrieve Windows telephony device names;
- A valid COM port string, formatted as *COMx*, where *x* is a valid COM port number. When you assign the 'Device' property with a COM port string, you bypass all Windows telephony intelligence, like dialing rules, port sharing and so on.

Windows telephony devices are highly recommended.



**Example:**

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 19200 bps Modem"         ' Use a Windows telephony device
...
```

**BaudRate property****Description:**

Baud rate at which the communications device operates. The default value is 0, which means that the baud rate setting is inherited from the Port/Device settings in the Control Panel of Windows. You should use the [UpdateCom](#) method if you want to change the baudrate and the port is already [opened](#).

This property can be one of the following values:

- 0 (Default - inherit baud rate from the device settings as defined in the Control Panel)
- 110
- 300
- 600
- 1200
- 2400
- 4800
- 9600
- 14400
- 19200
- 38400
- 56000
- 57600
- 115200
- 128000
- 230400
- 256000
- 460800
- 921800

**Example:**

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM2"                             ' Set device to COM2
objComport.BaudRate = 38400                             ' Use 38400 bps
objComport.Open
...
```

**Databits property****Description:**

Number of databits in a byte. The default value is [asDATABITS\\_DEFAULT](#), which means that the data bits setting is inherited from the Port/Device settings in the Control Panel of Windows. You cannot change the value when the port is already [opened](#).

**Example:**

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM2"                             ' Set device to COM2
objComport.DataBits = objComport.asDATABITS_7          ' Use 7 bit data bits
objComport.Open
...
```

**Stopbits property**

**Description:**

You can configure StopBits to be [asSTOPBITS\\_DEFAULT](#), [asSTOPBITS\\_1](#), [asSTOPBITS\\_2](#) or [asSTOPBITS\\_15](#).

If StopBits is [asSTOPBITS\\_1](#), one stop bit is used to indicate the end of data transmission.

If StopBits is [asSTOPBITS\\_2](#), two stop bits are used to indicate the end of data transmission.

If StopBits is [asSTOPBITS\\_15](#), the stop bit is transferred for 150% of the normal time used to transfer one bit.

The default value is [asSTOPBITS\\_DEFAULT](#), which means that the stop bits setting is inherited from the Port/Device settings in the Control Panel of Windows.

You cannot change the value when the port is already [opened](#).

**Example:**

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM2"                             ' Set device to COM2
objComport.StopBits = objComport.asSTOPBITS_15         ' Use 1.5 stop bits
objComport.Open
...
```

**Parity property****Description:**

Parity checking can detect errors of one bit only. An error in two bits might cause the data to have a seemingly valid parity, when in fact it is incorrect.

You can configure Parity to be none, odd, even, mark, or space.

If Parity is [asPARITY\\_DEFAULT](#), the parity setting is inherited from the Port/Device settings in the Control Panel of Windows.

If Parity is [asPARITY\\_NONE](#) (=none), parity checking is not performed and the parity bit is not transmitted.

If Parity is [asPARITY\\_ODD](#) (=odd), the number of mark bits (1's) in the data is counted, and the parity bit is asserted or unasserted to obtain an odd number of mark bits.

If Parity is [asPARITY\\_EVEN](#) (=even), the number of mark bits in the data is counted, and the parity bit is asserted or unasserted to obtain an even number of mark bits.

If Parity is [asPARITY\\_MARK](#) (=mark), the parity bit is asserted.

If Parity is [asPARITY\\_SPACE](#) (=space), the parity bit is unasserted.

The default value is [asPARITY\\_DEFAULT](#).

You cannot change the value when the port is already [opened](#).

**Example:**

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM2"                             ' Set device to COM2
objComport.Parity = objComport.asPARITY_NONE           ' No parity
objComport.Open
...
```

**HardwareFlowControl property****Description:**

Use Hardware Flow Control. The default value is [asFLOWCONTROL\\_DEFAULT](#), which means that the hardware flow control settings are inherited from the Port/Device settings in the Control Panel of Windows.

This property sets the generic hardware flow control control properties.

When you set 'HardwareFlowControl' to [asFLOWCONTROL\\_ENABLE](#):

- [DTRFlowControl](#) and [RTSFlowControl](#) are both set to [asFLOWCONTROL\\_ENABLE](#)
- [CTSFlowControl](#) and [DSRFlowControl](#) are both set to [asFLOWCONTROL\\_DISABLE](#)

When you set 'HardwareFlowControl' to [asFLOWCONTROL\\_DISABLE](#):

- [DTRFlowControl](#) and [RTSFlowControl](#) are both set to [asFLOWCONTROL\\_DISABLE](#)
- [CTSFlowControl](#) and [DSRFlowControl](#) are both set to [asFLOWCONTROL\\_DISABLE](#)

In most circumstances, it is not necessary to assign [DTRFlowControl](#), [RTSFlowControl](#), [CTSFlowControl](#) or [DSRFlowControl](#) individually. However, in some situations you want need to assign these properties individually.

#### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM2"                            ' Set device to COM2
objComport.HardwareFlowControl = objComport.asFLOWCONTROL_DISABLE ' No hardware flow control
objComport.Open
...
```

### **DTRFlowControl property**

#### Description:

Advanced Hardware Flow Control. Usually, the [HardwareFlowControl](#) property will suffice: it sets all four advanced hardware flow control flags ( [DTRFlowControl](#), [RTSFlowControl](#), [CTSFlowControl](#), [DSRFlowControl](#)). The default value is [asFLOWCONTROL\\_DEFAULT](#), which means that the DTR flow control settings are inherited from the Port/Device settings in the Control Panel of Windows.

You cannot change the value when the port is already [opened](#).

#### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM2"                            ' Set device to COM2
objComport.DTRFlowControl = objComport.asFLOWCONTROL_DISABLE ' No DTR hardware flow control
objComport.Open
...
```

### **RTSFlowControl property**

#### Description:

Advanced Hardware Flow Control. Usually, the [HardwareFlowControl](#) property will suffice: it sets all four advanced hardware flow control flags ( [DTRFlowControl](#), [RTSFlowControl](#), [CTSFlowControl](#), [DSRFlowControl](#)). The default value is [asFLOWCONTROL\\_DEFAULT](#), which means that the RTS flow control settings are inherited from the Port/Device settings in the Control Panel of Windows.

You cannot change the value when the port is already [opened](#).

#### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM2"                            ' Set device to COM2
objComport.RTSFlowControl = objComport.asFLOWCONTROL_DISABLE ' No RTS
objComport.Open
...
```

### **CTSFlowControl property**

#### Description:

Advanced Hardware Flow Control. Usually, the [HardwareFlowControl](#) property will suffice: it sets all four advanced hardware flow control flags ( [DTRFlowControl](#), [RTSFlowControl](#), [CTSFlowControl](#), [DSRFlowControl](#)).

The default value is [asFLOWCONTROL\\_DEFAULT](#), which means that the CTS flow control settings are inherited from the Port/Device settings in the Control Panel of Windows.

You cannot change the value when the port is already [opened](#).

#### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM2"                             ' Set device to COM2
objComport.CTSFlowControl = objComport.asFLOWCONTROL_DISABLE ' No CTS hardware flow control
objComport.Open
...
```

### DSRFlowControl property

#### Description:

Advanced Hardware Flow Control. Usually, the [HardwareFlowControl](#) property will suffice: it sets all four advanced hardware flow control flags ( [DTRFlowControl](#), [RTSFlowControl](#), [CTSFlowControl](#), [DSRFlowControl](#)). The default value is [asFLOWCONTROL\\_DEFAULT](#), which means that the DSR flow control settings are inherited from the Port/Device settings in the Control Panel of Windows.

You cannot change the value when the port is already [opened](#).

#### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM2"                             ' Set device to COM2
objComport.DSRFlowControl = objComport.asFLOWCONTROL_DISABLE ' No DSR hardware flow control
objComport.Open
...
```

### SoftwareFlowControl property

#### Description:

Software flow control. You should use the [UpdateCom](#) method if you want to change the baudrate after you have [opened](#) the port.

The default value is [asFLOWCONTROL\\_DEFAULT](#), which means that the software flow control settings are inherited from the Port/Device settings in the Control Panel of Windows. The other valid values are: [asFLOWCONTROL\\_DISABLE](#) and [asFLOWCONTROL\\_ENABLE](#).

#### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM2"                             ' Set device to COM2
objComport.SoftwareFlowControl = objComport.asFLOWCONTROL_ENABLE ' Use software flow control
objComport.Open
...
```

### ComTimeout property

#### Description:

Timeout of ReadString method, in milliseconds. You can call this method anytime you want. The default value is 1000.

#### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600bps Modem"           ' Set device to Standard 9600 bps Modem
str = objComport.ReadString                             ' Blocks until string read or 1s elapsed
WScript.Echo str
objComport.ComTimeout = 5000                           ' Set timeout to 5000 msec
```

```
str = objComport.ReadString           ' Blocks until string read or 5s elapsed
WScript.Echo str
objComport.Close                     ' Close port
...
```

## IsOpened property

### Description:

True if port is opened, otherwise False.

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")   ' Create new instance
objComport.Device = "Standard 9600 bps Modem"       ' Set device to Standard 9600 bps Modem
If ( objComport.IsOpened ) Then
    WScript.Echo "Port is opened"
    objComport.Close                               ' Close port
End If
```

## PreCommandDelay property

### Description:

Specifies a delay (in milliseconds) used before [WriteString](#) actually starts writing the command string. This property was introduced to support slow devices that do not accept a few commands right after each other. These devices need a small delay between commands, which can be accomplished by setting this 'PreCommandDelay' property.

Note that the property does NOT apply to the [WriteBytes](#) and [WriteByte](#) functions.

Default value: 0, indicating no delay between commands.

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")   ' Create new instance
objComport.Device = "Standard 9600 bps Modem"       ' Set device to Standard 9600 bps Modem
objComport.Open                                     ' Now open the port
If ( objComport.LastError = 0 ) Then
    objComport.PreCommandDelay = 500                ' WriteString will use delay of 500ms
                                                    ' before the command string is sent
    objComport.InterCharDelay = 10                  ' delay of 10 ms between each character
                                                    ' Transmitted by WriteString method
    ...
    objComport.WriteString "AT&F"
    ...
    objComport.Close                               ' Close the port
End If
```

## InterCharDelay property

### Description:

Specifies a delay (in milliseconds) used in [WriteString](#) between each character transmitted. This property was introduced to support slow devices that do not allow each character transmitted right after each other in command mode. These devices need a small delay between characters, which can be accomplished by setting this 'InterCharDelay' property.

Note that the property does NOT apply to the [WriteBytes](#) method.

Default value: 0, indicating no delay between characters.

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")   ' Create new instance
objComport.Device = "Standard 9600 bps Modem"       ' Set device to Standard 9600 bps Modem
objComport.Open                                     ' Now open the port
If ( objComport.LastError = 0 ) Then
```

```

objComport.PreCommandDelay = 500      ' WriteString will use delay of 500ms
                                       ' before the command string is sent
...
objComport.InterCharDelay  = 10      ' Delay of 10 ms between each character
                                       ' transmitted by WriteString method
objComport.WriteString "AT&F"
...
objComport.Close                    ' Close the port
End If

```

## NewLine property

### Description:

The character sequence that forms a newline. Default value: the CR (carriage return) string. Most frequent used newline strings:

- CR (carriage return) - a string containing only the ASCII-13 character (in C: "\r"; in VB: vbCr )
- LF (linefeed) - a string containing only the ASCII-10 character (in C: "\n"; in VB: vbLf )
- CRLF (carriage return / linefeed) - a string containing the ASCII-13, ASCII-10 sequence (in C: "\r\n"; in VB: vbCrLf )

A newline is used internally by two functions:

- [ReadString](#) - bytes are read from the port until a newline is detected;
- [WriteString](#) - bytes are written to the port. Finally a newline is sent.

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600bps Modem"           ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If ( objComport.LastError = 0 ) Then
    objComport.NewLine = vbCrLf                        ' Use CRLF in subsequent Read/WriteBytes calls
                                                         ' transmitted by WriteString method
    objComport.WriteString "AT&F"                    ' AT&F is sent, followed by a CRLF
    ...
    objComport.Close                                   ' Close the port
End If

```

## LogFile property

### Description:

By default, LogFile holds the empty string and nothing is logged. If you assign a valid file name to it, all device commands and responses will be written to this log file.

Output is always appended.

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.LogFile = "C:\MyLogFile.txt"                ' All operations are logged here
objComport.Device = "Standard 9600 bps Modem"           ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Open the port
...

```

## 5.2. Methods

### Methods

Method	Description
<a href="#">Clear</a>	Reset all properties to the default values

<a href="#">GetDeviceCount</a>	Return the number of Windows telephony devices installed on the local computer
<a href="#">GetDevice</a>	Retrieve a Windows telephony device name
<a href="#">Open</a>	Open a comport
<a href="#">Close</a>	Close a comport
<a href="#">ClearTX</a>	Clears the output buffer
<a href="#">ClearRX</a>	Clears the input buffer
<a href="#">ReadString</a>	Read an ASCII string from the comport
<a href="#">ReadByte</a>	Read a (binary) byte from the comport
<a href="#">ReadBytes</a>	Read a stream of (binary) bytes from the comport
<a href="#">WriteString</a>	Write an ASCII string to the comport
<a href="#">WriteByte</a>	Write a (binary) byte to the comport
<a href="#">WriteBytes</a>	Write a stream of (binary) bytes to the comport
<a href="#">UpdateCom</a>	Update the comport with new configuration settings
<a href="#">RaiseRTS</a>	Raise the RTS signal
<a href="#">RaiseDTR</a>	Raise the DTR signal
<a href="#">QueryCTS</a>	Query the CTS signal
<a href="#">QueryDSR</a>	Query the DSR signal
<a href="#">QueryDCD</a>	Query the DCD signal
<a href="#">QueryRI</a>	Query the RI signal
<a href="#">Sleep</a>	Be idle for some time
<a href="#">GetErrorDescription</a>	Get error description
<a href="#">SaveLicenseKey</a>	Save the License Key in the registry

## Clear method

### Description:

Reset all properties to their initial values.

### Parameters:

None.

### Return value:

Always 0.

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open
...
objComport.Close
objComport.Clear                                       ' Clear all properties
objComport.Device = "COM1"                             ' Set device to COM1:
objComport.Open
...
```

## GetDeviceCount method

### Description:

Returns the number of installed Windows telephony devices on the local computer.

**Parameters:**

- None

**Return value:**

The number of installed Windows telephony devices. Check the [LastError](#) property to see if the method was completed successfully.

NOTE: The number of Windows telephony devices does not include the number installed COM ports.

**Example:**

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
WScript.Echo "Number of installed Windows telephony devices: " & objComport.GetDeviceCount()
```

**GetDevice method****Description:**

Returns the n-th telephony device of the system. The number *n* can be between 0 and [GetDeviceCount\(\)](#)-1.

**Parameters:**

- Zero based index, to iterate over all telephony devices.

**Return value:**

The name of the device. Call the [LastError](#) method to see if the method was completed successfully. The name of the device can be assigned to the [Device](#) property to open a Windows telephony device.

**Example:**

```
Set objComport= CreateObject("AxSerial.ComPort")      ' Create new instance
n = objComport.GetDeviceCount()
For i = 0 to n-1
    WScript.Echo "Device " & i & ": " & objComport.GetDevice( i )
Next
```

**Example:**

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
If( objComport.GetDeviceCount() > 0 )
    objComport.Device = objComport.GetDevice( 0 )      ' Use the first telephony device
    objComport.WriteByte 32
End If
```

**Open method****Description:**

Open the comport. The [Device](#) indicates to port to open.

**Parameters:**

None.

**Return value:**

Always 0. Check [LastError](#) property to see if the method was completed successfully.

**Example:**



```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                       ' Now open the port
...

```

## Close method

### Description:

Close the COM port.

### Parameters:

None.

### Return value:

Always 0. Check [LastError](#) property to see if the method was completed successfully.

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                       ' Now open the port
If( objComport.LastError = 0 ) Then
    ...
    objComport.Close                                  ' Close the port
End If

```

## ClearTX method

### Description:

Clears the output buffer (if the device or UART has one).

### Parameters:

None.

### Return value:

Always 0. Check [LastError](#) property to see if the method was completed successfully.

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to COM2
objComport.Open                                       ' Now open the Standard 9600 bps Modem
If( objComport.LastError = 0 ) Then
    ...
    objComport.ClearTX                               ' Clear transmission queue
    objComport.WriteString "AT&F"
    ...
    objComport.Close                                  ' Close the port
End If

```

## ClearRX method

### Description:

Clears the input buffer (if the device or UART has one).

### Parameters:

None.

### Return value:

Always 0. Check [LastError](#) property to see if the method was completed successfully.

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If( objComport.LastError = 0 ) Then
    ...
    objComport.ClearTX                                 ' Clear transmission queue
    objComport.ClearRX                                 ' Clear transmission queue
    ...
    objComport.WriteString "AT&F"
    WScript.Echo objComport.ReadString                 ' Read incoming data
    ...
    objComport.Close                                   ' Close the port
End If
```

## ReadString method

### Description:

This method reads a string of data from the device.

The method returns when either a string, terminated by a [NewLine](#), is read from the device, or when the time specified by [ComTimeout](#) has elapsed.

In case of a timeout, the empty string will be returned and [LastError](#) will indicate a time-out.

### Parameters:

None.

### Return value:

The string that was read from the device.

Check [LastError](#) property to see if the method was completed successfully.

In case of a timeout, the empty string is returned, and [LastError](#) will indicate a time-out.

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If objComport.LastError = 0 Then
    objComport.ComTimeout = 2000
    str = objComport.ReadString                          ' Method will timeout after 2000 msec
    WScript.Echo "String read: " & str
    ...
    objComport.Close
End If
```

## ReadByte method

## Comport.WriteBytes method

### Description:

Send a stream of binary data to the device.

### Parameters:

- Stream of binary data

### Return value:

Always 0. Check [LastError](#) property to see if the method was completed successfully.

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If( objComport.LastError = 0 ) Then
...
data = Chr( 1 ) & Chr( 2 ) & Chr( 253 ) & Chr( 254 ) & Chr( 255 )
objComport.WriteBytes data                             ' Write stream of data
...
objComport.Close                                       ' Close the port
End If
```

## ReadBytes method

### Description:

This method sends a (binary) byte to the device.

### Parameters:

- One byte

### Return value:

Always 0. Check [LastError](#) property to see if the method was completed successfully.

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If( objComport.LastError = 0 ) Then
...
objComport.WriteByte 97                                ' [A]
objComport.WriteByte 116                              ' [T]
objComport.WriteByte 122                              ' [Z]
objComport.WriteByte 13                               ' [<CR>]; ATZ issued.
...
objComport.Close                                       ' Close the port
End If
```

## WriteString method

### Description:

This method sends a string of ASCII data to the device. Finally [NewLine](#) is sent to the device.

### Parameters:

- ASCII string to send

### Return value:

Always 0. Check [LastError](#) property to see if the method was completed successfully.

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If( objComport.LastError = 0 ) Then
    ...
    objComport.WriteString "AT&F"
    ...
    objComport.Close                                   ' Close the port
End If

```

## WriteByte method

### Description:

This method reads one byte of data from the device.

The method returns when either a byte is read from the device, or when the time specified by [ComTimeout](#) has elapsed. In case of a timeout, [LastError](#) will indicate a time-out.

### Parameters:

None.

### Return value:

The byte that was read from the device.

Check [LastError](#) property to see if the method was completed successfully.

In case of a timeout, the [LastError](#) will be set to indicate a timeout.

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to COM2
objComport.Open                                         ' Now open the port
If objComport.LastError = 0 Then
    objComport.ComTimeout = 2000
    ..
    bt = objComport.ReadByte
    If( objComport.LastError = 0 ) Then                  ' Method will timeout after 2000 msec
        WScript.Echo "Byte read: " & bt
    End If
    ...
    objComport.Close
End If

```

## WriteBytes method

### Description:

This method reads a stream of binary data from the device.

The method returns when when the time specified by [ComTimeout](#) has elapsed.

### Parameters:

None.

### Return value:

The method returns when when the time specified by [ComTimeout](#) has elapsed.

If data was read from the port, 0 will be returned; in case of a timeout without data being read from the port, the [LastError](#) property will indicate a timeout.

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance

```

```

objComport.Device = "Standard 9600 bps Modem"      ' Set device to Standard 9600 bps Modem
objComport.Open                                   ' Now open the port
If objComport.LastError = 0 Then
    objComport.ComTimeout = 2000
    ...
    btData = objComport.ReadBytes                  ' Read stream of binary data
    If ( objComport.LastError = 0 ) Then           ' Method will timeout after 2000 msec
        WScript.Echo "Number of bytes read: " & Len( btData )
    End If
    ...
    objComport.Close
End If

```

## UpdateCom method

### Description:

If a COM port is already opened and you changed the [baudrate](#) or [Software Flow Control](#), you must this method to let the changes take effect. of the comport.

### Parameters:

None.

### Return value:

The method returns when when the time specified by [ComTimeout](#) has elapsed.

If data was read from the port, 0 will be returned; in case of a timeout without data being read from the port, the [LastError](#) property will indicate a timeout.

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")  ' Create new instance
objComport.Device = "Standard 9600 bps Modem"      ' Set device to Standard 9600 bps Modem
objComport.Open                                   ' Now open the port
If objComport.LastError = 0 Then
    objComport.BaudRate = 57600                    ' Use different baudrate
    objComport.SoftwareFlowControl = asFLOWCONTROL_ENABLE ' Use software flow control
    objComport.UpdateCom                           ' Update COM port
    ...
    objComport.Close
End If

```

## RaiseRTS method

### Comport.RaiseRTS method

### Description:

Raise (or lower) the RTS (Request-To-Send) signal. Raising this signal has nothing to do with the [HardwareFlowControl](#) property: 'RaiseRTS' just raises (or lowers) the RTS signal, regardless of the hardware flow control used.

### Parameters:

Boolean: True to raise the signal, or False to lower the signal.

### Return value:

Always 0. Check [LastError](#) property to see if the method was completed successfully.

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If objComport.LastError = 0 Then
    objComport.RaiseRTS( True )                        ' Raise RTS
    ...
    objComport.RaiseRTS( False )                      ' Lower RTS
    ...
    objComport.Close
End If

```

## RaisedDTR method

### Description:

Raise (or lower) the DTR (Data-Terminal-Ready) signal. Raising this signal has nothing to do with the [HardwareFlowControl](#) property: 'RaisedDTR' just raises (or lowers) the DTR signal, regardless of the hardware flow control used.

### Parameters:

- Boolean: True to raise the signal, or False to lower the signal.

### Return value:

Always 0. Check [LastError](#) property to see if the method was completed successfully.

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If objComport.LastError = 0 Then
    objComport.RaisedDTR( True )                      ' Raise DTR
    ...
    objComport.RaisedDTR( False )                    ' Lower DTR
    ...
    objComport.Close
End If

```

## QueryCTS method

### Description:

Query the actual value of the CTS (Clear-To-Send) signal.

### Parameters:

None.

### Return value:

True if CTS is raised, otherwise: False

### Example:

```

Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If objComport.LastError = 0 Then
    ...
    bVal = objComport.QueryCTS()                      ' Query CTS signal
    WScript.Echo "CTS: " & bVal
    ...
    objComport.Close
End If

```

## QueryDSR method

### Description:

Query the actual value of the DSR (Data-Set-Ready) signal.

### Parameters:

None.

### **Return value:**

True if DSR is raised, otherwise: False

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If objComport.LastError = 0 Then
    ...
    bVal = objComport.QueryDSR()                        ' Query DSR signal
    WScript.Echo "DSR: " & bVal
    ...
    objComport.Close
End If
```

## QueryDCD method

### Description:

Query the actual value of the DCD (Data-Carrier-Detect) signal.

### Parameters:

None.

### **Return value:**

True if DCD is raised, otherwise: False

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                         ' Now open the port
If objComport.LastError = 0 Then
    ...
    bVal = objComport.QueryDCD()                        ' Query DCD signal
    WScript.Echo "DCD: " & bVal
    ...
    objComport.Close
End If
```

## QueryRI method

### Description:

Query the actual value of the RI (Ring-Indicator) signal.

### Parameters:

None.

**Return value:**

True if RI is raised, otherwise: False

**Example:**

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                       ' Now open the port
If objComport.LastError = 0 Then
    ...
    bVal = objComport.QueryRI()                      ' Query RI signal
    WScript.Echo "RI: " & bVal
    ...
    objComport.Close
End If
```

**Sleep method****Comport.Sleep method****Description:**

This method can be used in your script anywhere you want; it will suspend the program. One paramter is required: the number of milliseconds you want to suspend.

**Parameters:**

- Number of milliseonds to suspend

**Return value:**

Always 0. Check [LastError](#) property to see if the method was completed successfully.

**Example:**

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "Standard 9600 bps Modem"          ' Set device to Standard 9600 bps Modem
objComport.Open                                       ' Now open the port
If objComport.LastError = 0 Then
    objComport.WriteString( "AT&F" )                 ' Suspend for 1000 msecs, then resume
    objComport.Sleep( 1000 )
    WScript.Echo objComport.ReadString
    objComport.Close
End If
```

**GetErrorDescription method****Comport.GetErrorDescription method****Description:**

GetErrorDescription provides the error description of a given error code.

**Parameters:**

- Error code.

**Return value:**

The error description that is associated with the given error code.

**Example:**



```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.Device = "COM9"                            ' Set device to COM9
...
WScript.Echo "LastError: " & objComport.LastError
WScript.Echo "Error description: " & objComport.GetErrorDescription( objComport.LastError )
...
```

## SaveLicenseKey method

### Description:

Use SaveLicenseKey to store the license key permanently in the registry. When a license key is saved, it is restored automatically every time a new instance of the object ('ComPort') is created. It is not recommended to save the license key if you distribute the component with your own software, because the key can be easily used by others.

### Parameters:

- None.

### Return value:

Always 0. Check [LastError](#) property to see if the method was completed successfully.

### Example:

```
Set objComport = CreateObject("AxSerial.ComPort")      ' Create new instance
objComport.LicenseKey = "XXXXXX-XXXXXX-XXXXXX"
objComport.SaveLicenseKey                             ' Save license key to registry
```

For more information, see [Product Activation](#).

## 5.3. Flags

### Flow Control definitions

Constant	Value	Description
asFLOWCONTROL_DEFAULT	0	Flow Control setting is inherited from the Port/Device settings in the Control Panel of Windows.
asFLOWCONTROL_DISABLE	1	Disable Flow Control
asFLOWCONTROL_ENABLE	2	Enable Flow Control

### Data Bit Flags definitions

Constant	Value	Description
asDATABITS_DEFAULT	0	Stop bits setting is inherited from the Port/Device settings in the Control Panel of Windows.
asDATABITS_7	7	7 data bits
asSTOPBITS_8	8	8 data bits

### Stop Bit Flags definitions

Constant	Value	Description
asSTOPBITS_DEFAULT	0	Stop bits setting is inherited from the Port/Device settings in the Control Panel of Windows.
asSTOPBITS_1	1	1 bit stop bit (default)
asSTOPBITS_2	2	2 bit stop bit
asSTOPBITS_15	15	1.5 bit parity

## Parity Codes definitions

Constant	Value	Description
asPARITY_DEFAULT	0	No parity bits
asPARITY_NONE	1	No parity bits
asPARITY_ODD	2	Odd parity
asPARITY_EVEN	3	Even parity bits
asPARITY_MARK	4	Mark parity
asPARITY_SPACE	5	Space parity

## 6. Error Codes

### 6.1. Introduction

When a method is called, the result of the method is stored in the object's [LastError](#) property. When [LastError](#) is 0, it means that the last called method completed successfully; otherwise, an error occurred.

The value of the LastError tells you why the method failed. All error codes are listed on the ActiveXperts web site:

[www.activexperts.com/support/errorcodes](http://www.activexperts.com/support/errorcodes) (list of error codes).

Here, you can also lookup a specific error to find its description.

You can also call the [GetErrorDescription](#) method to find the error description.

## 7. Samples

### 7.1. Introduction

Samples for Visual Basic, Visual Basic .NET, Visual C++, Visual C# .NET, ASP and VBScript are included as part of the installation. You can also find the samples on our website at [ftp.activexperts-labs.com/samples/serial-port-component](http://ftp.activexperts-labs.com/samples/serial-port-component).

## 8. Troubleshooting

### 8.1. FAQ's

Visit our website for a complete list of FAQ's at: <http://www.activexperts.com/support>

### 8.2. Contact us

Please contact our website for support questions about this product, or send an email to our support-staff:

Website: <http://www.activexperts.com/support>

E-mail: [support@activexperts.com](mailto:support@activexperts.com)

## 9. Purchase and Product Activation

### 9.1. Purchase

Please visit [www.activexperts.com/sales](http://www.activexperts.com/sales) to buy the product. Here, you can also find the latest prices.

You can also contact us via email: [sales@activexperts.com](mailto:sales@activexperts.com)

After purchasing the product, you will receive one or more product registration keys.

### 9.2. Product Activation

After purchasing the product, you will receive a license key.

There are four ways to activate (unlock) the component using this license key:

#### 1. Directly from your program code

You can unlock the component by using the [LicenseKey](#) property. This way, the license is NOT stored in the registry of the computer. This is the recommended way when distributing this component with your own software.

#### 2. Store the license key in the registry - Installation

When the license key is entered during Setup (AxSerialSetup.exe, available from the ActiveXperts [download site](#)), the license key will be saved in the following registry key: 'HKEY\_LOCAL\_MACHINE\Software\ActiveXperts\Serial Port Component\LicenseKey'

Once the license key is stored in the registry, the [LicenseKey](#) property will be assigned automatically with that value each time the object is instantiated.

#### 3. Store the license key in the registry - Manually

You can enter the license key manually (e.g. through REGEDIT.EXE) in following registry key: 'HKEY\_LOCAL\_MACHINE\Software\ActiveXperts\Serial Port Component\LicenseKey'

Once the license key is stored in the registry, the [LicenseKey](#) property will be assigned automatically with that value each time the object is instantiated.

#### 4. Store the license key in the registry - SaveLicenseKey method

You can enter the license key by calling the [SaveLicenseKey](#) method. You need to call [SaveLicenseKey](#) only once.

```
Set objComport = CreateObject("AxSerial.ComPort") ' Create new instance
objComport.LicenseKey = "XXXXXX-XXXXXX-XXXXXX" ' Replace XXXXX-XXXXXX-XXXXXX by your own key
objComport.SaveLicenseKey
```

Once the license key is stored in the registry, the [LicenseKey](#) property will be assigned automatically with that value each time the object is instantiated.

### 9.3. Distribution License

For information about how to use the registration code with a Distribution License, please read the following document: [How to distribute an ActiveXperts Component](#).

## Appendix A License Agreement

PLEASE READ THIS SOFTWARE LICENSE AGREEMENT CAREFULLY BEFORE DOWNLOADING OR USING THE SOFTWARE. BY CLICKING ON THE "ACCEPT" BUTTON, OPENING THE PACKAGE, DOWNLOADING THE PRODUCT, OR USING THE EQUIPMENT THAT CONTAINS THIS PRODUCT, YOU ARE CONSENTING TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, CLICK THE "DO NOT ACCEPT" BUTTON AND THE INSTALLATION PROCESS WILL NOT CONTINUE, RETURN THE PRODUCT TO THE PLACE OF PURCHASE FOR A FULL REFUND, OR DO NOT DOWNLOAD THE PRODUCT.

#### GENERAL

In this Software License Agreement:

- (i) "ActiveXperts" means ActiveXperts Software B.V.
- (ii) "Customer" means the individual(s), organization or business entity buying a license of the Software from ActiveXperts or its Distributors or its Resellers.
- (iii) "Software" means computer programs (and their storage medium) supplied by ActiveXperts and known collectively as "ActiveXperts Serial Port Component" in which ActiveXperts has property rights and any user manuals, operating instructions, brochures and all other documentation relating to the said computer programs (the expression "Software" to include all or any part or any combination of Software).

#### 1. LICENSE GRANT

ActiveXperts grants Customer the following rights provided that you comply with all terms and conditions of this License Agreement:

- (a) Installation and use. Customer may install, use, access, display and run one copy of the Software on a single computer, such as a workstation, terminal or other device ("Workstation Computer"). A "License Pack" allows you to install, use, access, display and run additional copies of the Software up to the number of "Licensed Copies"

specified above.

(b) Reservation of Rights. ActiveXperts reserves all rights not expressly granted to you in this License Agreement.

## 2. UPGRADES AND SUPPLEMENTS

To use a product identified as an upgrade, you must first be licensed for the Software as eligible for the upgrade. After upgrading, Customer may no longer use the product that formed the basis for Customer's upgrade eligibility.

This License Agreement applies to updates or supplements to the original Software provided by ActiveXperts, unless we provide other terms along with the update or supplement.

## 3. LIMITATION ON REVERSE ENGINEERING, DECOMPILATION, AND DISASSEMBLY

Customer may not reverse engineer, decompile, or disassemble the Software, except and only to the extent that it is expressly permitted by applicable law notwithstanding this limitation.

## 4. TERMINATION

Without prejudice to any other rights, ActiveXperts may cancel this License Agreement if Customer does not abide by the terms and conditions of this License Agreement, in which case you must destroy all copies of the Software and all of its component parts.

## 5. NOT FOR RESALE SOFTWARE

Software identified as "Not for Resale" or "NFR," may not be resold, transferred or used for any purpose other than demonstration, test or evaluation.

## 6. LIMITED WARRANTY

ActiveXperts warrants that for a period of ninety (90) days from the date of shipment from ActiveXperts: (i) the media on which the Software is furnished will be free of defects in materials and workmanship under normal use; and (ii) the Software substantially conforms to its published specifications. Except for the foregoing, the Software is provided AS IS. This limited warranty extends only to Customer as the original licensee. Customer's exclusive remedy and the entire liability of ActiveXperts and its suppliers under this limited warranty will be, at ActiveXperts or its service center's option, repair, replacement, or refund of the Software if reported (or, upon request, returned) to the party supplying the Software to Customer. In no event does ActiveXperts warrant that the Software is error free or that Customer will be able to operate the Software without problems or interruptions.

This warranty does not apply if the software (a) has been altered, except by ActiveXperts, (b) has not been installed, operated, repaired, or maintained in accordance with instructions supplied by ActiveXperts, (c) has been subjected to abnormal physical or electrical stress, misuse, negligence, or accident, or (d) is used in ultrahazardous activities.

## 7. LIMITATION OF LIABILITY AND REMEDIES.

Notwithstanding any damages that you might incur for any reason whatsoever (including, without limitation, all damages referenced above and all direct or general damages), the entire liability of ActiveXperts and any of its suppliers under any provision of this License Agreement

and your exclusive remedy for all of the foregoing (except for any remedy of repair or replacement elected by ActiveXperts with respect to any breach of the Limited Warranty) shall be limited to the greater of the amount actually paid by you for the Software or U.S.\$5.00. The foregoing limitations, exclusions and disclaimers (including Sections 4, 5 and 6 above) shall apply to the maximum extent permitted by applicable law, even if any remedy fails its essential purpose.

#### 8. ENTIRE AGREEMENT

This License Agreement (including any addendum or amendment to this License Agreements which is included with the Software) are the entire agreement between you and ActiveXperts relating to the Software and the support services (if any) and they supersede all prior or contemporaneous oral or written communications, proposals and representations with respect to the Software or any other subject matter covered by this License Agreement. To the extent the terms of any ActiveXperts policies or programs for support services conflict with the terms of this License Agreement, the terms of this License Agreement shall control.

This Agreement shall be construed in accordance with the laws of The Netherlands and the Dutch courts shall have sole jurisdiction in any dispute relating to these conditions. If any part of these conditions shall be or become invalid or unenforceable in any way and to any extent by any existing or future rule of law, order, statute or regulation applicable thereto, then the same shall to the extent of such invalidity or enforceability be deemed to have been deleted from the conditions which shall remain in full force and effect as regards all other provisions.

#### 9. Copyright

The Software is protected by copyright and other intellectual property laws and treaties. ActiveXperts or its suppliers own the title, copyright, and other intellectual property rights in the Software. The Software is licensed, not sold.