Symantec pcAnywhere™ OLE Automation Guide

Symantec pcAnywhere™

Symantec pcAnywhere OLE Automation Guide

Documentation version 10.0

Copyright Notice

Trademarks

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

C O N T E N T S

Chapter 1 Using OLE Automation

Chapter 2 C++ Object Definitions

Chapter 3 Visual Basic Object Definitions

CHAPTER 1

Using OLE Automation

Introduction

pcAnywhere's Automation Server is an application that lets external applications manage Host and Remote information files in pcAnywhere. The Automation Server also lets external applications manage the connection to a Host and file transfer (to and from the Host).

This document contains several examples, written in both Visual Basic and C++, that illustrate the way to connect to, and use, the pcAnywhere Automation Server.

For example, users often want to distribute files to a number of computers on the company's network. Using the Automation Server, pcAnywhere can be automated so that it reads a location from a list of computers, connects to the pcAnywhere Host on that computer, transfers files, and continues through the rest of the computer list.

Overview of technology

OLE Automation is the technology that makes the pcAnywhere Automation Server possible. An external application accessing an Automation server does so by first connecting to the server and then requesting access to one or more of its published interfaces.

An interface is an entry point that allows access to one or more related methods or properties. Once an application obtains an interface on the server, it proceeds to call any interface methods, as though they were part of the external application.

The pcAnywhere Automation Server is an executable file named Winawsvr.exe. Although the Winawsvr application is not part of pcAnywhere, it uses pcAnywhere internals to perform its tasks.

Think of the Automation Server as a programmable replacement for the pcAnywhere user interface. The behavior seen when using the pcAnywhere interface is the same behavior seen when using the Automation Server. When you create a Host object in pcAnywhere, the first available TAPI device is assigned by default. Similarly, when you create a Host object via the Automation Server and then enumerate through the list of assigned connections, the first available TAPI device is already assigned.

Implementing the technology

When connected to the Automation Server and its interfaces, identifier parameters, also known as Globally Unique Identifiers (GUIDs), are passed to the automation library API functions. There is a single GUID representing the Automation Server itself and a GUID for each of the server's exposed interfaces. In order for the application to succeed in connecting to the server and its interfaces, these GUIDs must be present in the registry.

When using a computer with pcAnywhere installed on it, the GUID entries have already been added to the registry to allow the user to start using the Automation Server immediately. If there is a need to run the application on a computer that does not have an installed version of pcAnywhere, run Winawsvr.exe once as an application. Doing this registers the necessary GUIDs on the computer.

Accessing the automation server

You can access the pcAnywhere Automation Server via any language platform that supports OLE Automation. The two most popular language platforms that support OLE automation are Microsoft Visual C++ and Microsoft Visual Basic.

The coding principles for these two platforms are similar, although in the Visual Basic environment much of the low-level work is performed behind the scenes by the Visual Basic run-time system.

Using Microsoft Visual C++

The following procedure gives the general steps to create a C++ application that accesses the pcAnywhere automation server.

To create a C++ application

1. In Visual C++, create an MFC application.
2. On the View menu, click **Class Wizard**.
3. Click **Add Class > From a type library**.
4. Select **winawsvr.tlb**.
5. In the Confirm Classes dialog box, click **OK** to import all class definitions.
6. Click **Add Class > From a type library**.
7. Select **awrem32.tlb**.
8. In the Confirm Classes dialog box, click **OK** to import all class definitions.
9. In the Class Wizard dialog box, click **OK** to complete the import.

   The classes are added to the application. These classes allow you to manipulate objects and manage connections.

To view the added classes

- Click the **ClassView** tab.

All of the Automation Server's interfaces and methods are exposed through these classes. In general, the data manager classes provide the functionality needed to obtain an interface to the Automation Server and perform a number of useful high-level operations on the interface's associated object type.

Use the data manager object to determine or change the current directory, enumerate through the list of data object files in the current directory, and create, retrieve, or delete a named object. Once created or retrieved, an object uses the associated data object class to examine or modify any of its exposed properties. Most of these properties are exposed through a pair of methods that begin with the word Get or Set. For example, a user calls the GetPhoneNumber method to examine the object's current phone number property, and calls SetPhoneNumber to set it.

In addition to the classes, the application now has four new files: Winawsvr.h, Winawsvr.cpp, Awrem32.h, and Awrem32.cpp.

To view the added files

- Click the **FileView** tab.

The added files contain the Automation Server's class definitions and implementations. There is no need to edit these files, but each of the application source files containing calls to the Automation Server's methods must include Winawsvr.h.

For more information, see "C++ Object Definitions" on page 11.

Using Microsoft Visual Basic

Visual Basic can interact with OLE automation servers. When you create a Standard Exe project and enter code to access the Automation Server, Visual Basic takes the high-level method calls in the source and expands them internally into the corresponding low-level OLE automation method calls.

The following procedure gives the general steps to use Visual Basic to accesses the pcAnywhere automation server.

To use Visual Basic to access the Automation server

1. Add a pair of Object variables for each of the pcAnywhere objects you intend to access.

   For example, when working with Remote objects, DIM a RemoteDataManager and a RemoteDataObject as Object.

2. Use the RemoteDataManager to attach to the Remote object's data manager.

   For example, call the CreateObject method with WINAWSVR.REMOTEDATAMANAGER as a parameter.

   Visual Basic uses the textual parameter to locate the manager's identifier in the registry and returns the interface to that manager.

3. Once there is a valid data manager object, use it to determine the current directory, change to another directory, enumerate the associated data object files in the current directory, or create, retrieve, or delete a data object file.

4. After a data object is created or retrieved, you can get or set properties of the object.

   The Visual Basic syntax does not use a property's name to differentiate between getting and setting its value. Instead, the property's position

in relation to the assignment operator determines whether the underlying method call is a Get or a Set.

The following examples demonstrate a Get and a Set:

- To get an object's phone number value, place the property name to the right of the assignment operator.

  For example, s = RemoteData.PhoneNumber(), where s is a string variable.

- To change the current phone number, place the property name to the left of the assignment operator.

  For example, RemoteData.PhoneNumber = "555-1212"

For more information, see "Visual Basic Object Definitions" on page 45.

References

- Blaszczak, Mike. 1997. *Professional MFC with Visual C++ 5.* Birmingham, UK.: Wrox Press.
- Box, Don. 1998 *Essential COM.* Reading, Mass.: Addison-Wesley.
- Brockschmidt, Kraig. 1995. *Inside OLE, Second Edition*. Redmond, Wash.: Microsoft Press.
- Horton, Ivor. 1997. *Beginning MFC Programming*. Birmingham, UK.: Wrox Press.
- Rogerson, Dale. 1997. *Inside COM.* Redmond, Wash.: Microsoft Press.
- Templeman, Julian. 1997. *Beginning MFC COM Programming.* Birmingham, UK.: Wrox Press.

CHAPTER 2

C++ Object Definitions

This section describes the C++ objects that support the pcAnywhere WINAWSVR OLE Objects and the AWREM32 OLE Objects. The objects for WINAWSVR are CRemoteDataManager, CRemoteData, CHostDataManager, and CHostData. AWREM32 has one object.

- **CremoteDataManager:** Provides methods to create, open, modify, save, and delete CRemoteData objects. CRemoteData defines the pcAnywhere parameters that drive the pcAnywhere Remote for a remote control session.
- **ChostDataManager:** Provides methods to create, open, modify, save, and delete CHostData objects. CHostData defines the pcAnywhere parameters that drive the pcAnywhere Host for a remote control session.
- **AWREM32:** Eight interfaces are provided that allow for connecting, disconnecting, file transfer, connection status, creating folders, running programs on the host, and retrieving error status.

Some functions, including Gateways, are not available in pcAnywhere 10.0, but are included here for use with previous versions.

For functions involving passwords, password values can be set, but not retrieved. This is for security purposes.

CRemoteDataManager

Methods

BSTR CurrentDirectory0;

Returns the full path name of the current directory in which pcAnywhere Remote objects are stored.

Return Value	
BSTR	The full path name of the current pcAnywhere data directory.

BOOL ChangeDirectory(LPCTSTR lpszNewDirectory);

Changes the current directory in which pcAnywhere Remote objects are stored.

Parameters	
LPCTSTR lpszNewDirectory	Name of an existing directory.
Return Value	
BOOL	True if successful.

BOOL FindFirst(LPCTSTR lpszPattern, BSTR FAR* pbstrFullQualName);

Finds the first pcAnywhere Remote object file (*.CHF) in the current directory, based on the specified file name pattern.

Parameters	
LPCTSTR lpszPattern	File name pattern to filter object files ("*" finds all Remote object files in the current directory).
BSTR FAR * pbstrFullQualName	Return buffer for full path name of the Remote object file matching the specified pattern.

Return Value	
BOOL	True if a Remote object file matching the specified pattern is found. The full path name of the matching file is stored in pbstrFullQualName.

BOOL FindNext(BSTR FAR* pbstrFullQualName);

After FindFirst()has been successfully called to get the name of a Remote object file in the current directory, FindNext() can be called to find the next file matching the pattern, if any.

Parameters	
BSTR FAR * pbstrFullQualName	Return buffer for full path name of the Remote object file matching the pattern specified in the original call to FindFirst().
Return Value	
BOOL	True if another Remote object file matching the pattern specified in the call to FindFirst() is found. The full path name of the matching file is stored in pbstrFullQualName.

LPDISPATCH RetrieveObject(LPCTSTR lpszFQName, short wAccessMode, LPCTSTR lpszPassword);

Retrieves a CRemoteData object by file name.

Parameters	
LPCTSTR lpszFQName	The fully qualified Remote object file name to be loaded.

short wAccessMode	Specifies how this object is to be used. This is related to the password protection. The options include: 0 = Not specified 1 = View only 2 = View and Modify 3 = Execute
LPCTSTR lpszPassword	Object password. May be NULL.

LPDISPATCH RetrieveObjectEx(LPCTSTR lpszFQName, short wAccessMode, LPCTSTR lpszPassword);

Retrieves a CRemoteDataEx object by file name.

Parameters	
LPCTSTR lpszFQName	The fully qualified Remote object file name to be loaded.
short wAccessMode	Specifies how this object is to be used. This is related to the password protection. The options include: 0 = Not specified 1 = View only 2 = View and Modify 3 = Execute
LPCTSTR lpszPassword	Object password. May be NULL.
Return Value	
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CRemoteDataEx object. The sample code at the end of this section illustrates how to attach this pointer to a CRemoteDataEx object.

LPDISPATCH CreateObject(LPCTSTR lpszName);

Creates a CRemoteData object and returns an LPDISPATCH pointer to it.

Parameters	
LPCTSTR lpszFQName	The fully qualified Remote object file name for new object.
Return Value	
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CRemoteData object. The sample code at the end of this section illustrates how to attach this pointer to a CRemoteData object.

LPDISPATCH CreateObjectEx(LPCTSTR lpszName);

Creates a CRemoteDataEx object and returns an LPDISPATCH pointer to it.

Parameters	
LPCTSTR lpszFQName	The fully qualified Remote object file name for new object.
Return Value	
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CRemoteDataEx object. The sample code at the end of this section illustrates how to attach this pointer to a CRemoteDataEx object.

BOOL DeleteObject(LPCTSTR lpszFQName, LPCTSTR lpszPassword);

Deletes a remote object file.

Parameters	
LPCTSTR lpszFQName	The fully qualified Remote object file name of the object to be deleted.
LPCTSTR lpszPassword	Object password. May be NULL.
Return Value	
BOOL	True if object is deleted.

BOOL Launch(LPCTSTR lpszFQName);

Launches a Remote object file. This opens the pcAnywhere Remote terminal window.

Parameters	
LPCTSTR lpszFQName	The fully qualified Remote object file of object to be launched.
Return Value	
BOOL	True if object is successfully launched.

CRemoteData

Use this object to modify Remote object data.

Get and Set Methods

The following methods are used to get and set properties of the CRemoteData object.

```
BSTR GetComputerName();
void SetComputerName(LPCTSTR lpszNewValue);
```

The computer name is the name of the pcAnywhere Host computer to be called when the Remote object is launched.

```
BSTR GetPhoneNumber();
void SetPhoneNumber(LPCTSTR lpszNewValue);
```

The phone number is the number to dial to establish a modem connection to a pcAnywhere Host computer.

```
BOOL GetUseDialingProperties();
void SetUseDialingProperties(BOOL bNewValue);
```

Indicates whether TAPI dialing properties should be used (location information) (TRUE), or the phone number string should be used exactly as it appears (FALSE).

```
BSTR GetAreaCode();
void SetAreaCode(LPCTSTR lpszNewValue);
```

If dialing properties are to be used, this is the area code of the number to be called.

```
BSTR GetCountryCode();
void SetCountryCode(LPCTSTR lpszNewValue);
```

If dialing properties are to be used, this is the country code of the number to be called.

```
short GetRedialCount();
void SetRedialCount(short nNewValue);
```

The number of times to retry dialing this number if the call fails.

```
short GetRedialDelay();
void SetRedialDelay(short nNewValue);
```

The time to wait (in seconds) between redial attempts.

```
BSTR GetAutoLoginName();
void SetAutoLoginName(LPCTSTR lpszNewValue);
```

The login name to be sent to the Host when a connection is made. If this is left empty, the user is prompted for a login name on connection.

```
BSTR GetAutoLoginPassword();
void SetAutoLoginPassword(LPCTSTR lpszNewValue);
```

The login password to be sent to the Host when a connection is made. If this is left empty, the user is prompted for a password on connection.

```
BSTR GetPassword();
void SetPassword(LPCTSTR lpszNewValue);
```

The password for this object.

```
BOOL GetExecuteProtection();
void SetExecuteProtection(BOOL bNewValue);
```

The object can only be launched if the password is used (TRUE).

```
BOOL GetReadProtection();
void SetReadProtection(BOOL bNewValue);
```

The object can only be viewed if the correct password is provided (TRUE).

```
BOOL GetWriteProtection();
void SetWriteProtection(BOOL bNewValue);
```

The object can only be written if the correct password is provided (TRUE).

```
BOOL GetLogSession();
void SetLogSession(BOOL bNewValue);
```

Controls whether sessions using this object are logged.

```
BOOL GetRecordSession();
void SetRecordSession(BOOL bNewValue);
```

Controls whether sessions using this object are recorded from the beginning.

```
BSTR GetRecordFile();
void SetRecordFile(LPCTSTR lpszNewValue);
```

The name of the record file for sessions using this object.

```
BOOL GetRunOnConnect();
void SetRunOnConnect(BOOL bNewValue);
```

Run one of the following procedures on connection (TRUE).

```
BSTR GetScriptFile();
void SetScriptFile(LPCTSTR lpszNewValue);
```

Script to run on connection (if RunOnConnect is TRUE).

```
BSTR GetAutoXferFile();
void SetAutoXferFile(LPCTSTR lpszNewValue);
```

AutoXfer commands to run on connection (if RunOnConnect is TRUE).

```
BSTR GetConnectionType();
void SetConnectionType(LPCTSTR lpszNewValue);
```

The available connection types include:

- COM1
- COM2
- COM3
- COM4
- SPX
- NetBIOS
- TCP/IP
- LPT1
- LPT2
- LPT3
- LPT4
- ISDN via CAPI 2.0
- Infrared
- DEFAULT TAPI

Also, the name of a TAPI device can be used as a connection type. “DEFAULT TAPI” uses the first TAPI device found in the system. To use a specific TAPI device, use FirstConnectionType() / NextConnectionType() to search for available devices.

Detail Methods

These are methods used to get and set details for each of the connection types. When a Remote object is assigned a connection type, the device details are set to valid default values.

COM device details

```
BSTR GetComParity();
void SetComParity(LPCTSTR lpszNewValue);
```

Values include:

- None
- Odd
- Even
- Mark
- Space

```
BSTR GetComFlowControl();
void SetComFlowControl(LPCTSTR lpszNewValue);
```

Values include:

- <None>
- XONXOFF
- RTS/CTS
- BOTH

```
BSTR GetComStartedBy();
void SetComStartedBy(LPCTSTR lpszNewValue);
```

Values include:

- Always connected
- Carrier detect (DCD)
- Clear to send (CTS)
- Data set ready (DSR)
- Ring indicator (RI)
- Receive 2 <CR>'s
- Modem response

```
BSTR GetComEndedBy();
void SetComEndedBy(LPCTSTR lpszNewValue);
```

Values include:

- Always connected
- Carrier detect (DCD)
- Clear to send (CTS)
- Data set ready (DSR)
- Ring indicator (RI)

```
long GetComSpeed();
void SetComSpeed(long nNewValue);
```

Values include:

- 110
- 300
- 600
- 1200
- 2400
- 4800
- 9600
- 19200
- 38400
- 57600
- 115200

Network (TCP/IP, SPX, Banyan) device details

```
BOOL GetGatewayUse();
void SetGatewayUse(BOOL bNewValue);
```

Connect through a pcAnywhere Gateway (TRUE).

```
BSTR GetGatewayName();
void SetGatewayName(LPCTSTR lpszNewValue);
```

Name of pcAnywhere Gateway to use.

```
BSTR GetGatewayClass();
void SetGatewayClass(LPCTSTR lpszNewValue);
```

Class of pcAnywhere Gateway to use.

```
BSTR GetGatewayParity();
void SetGatewayParity(LPCTSTR lpszNewValue);
```

Values include:

- <None>
- Odd

- Even
- Mark
- Space

NetBios device details

```
short GetLanaNumber();
void SetLanaNumber(short nNewValue);
```

The LANA (LAN Adapter) number to use for this connection.

ISDN via CAPI 2.0 device details

```
BOOL GetCapiChannelBonding();
void SetCapiChannelBonding(BOOL bNewValue);
```

Use Channel Bonding (uses 2 ISDN channels for one connection) (TRUE).

```
BSTR GetCapiExtensions();
void SetCapiExtensions(LPCTSTR lpszNewValue);
```

Methods

The following are the normal methods of the object. They are not used to get and set properties.

short ConnectionTypes0;

Returns the number of connection types available.

Return Value	
Short	The number of connection types actually available on this system.

BSTR FirstConnectionType0; and BSTR NextConnectionType0;

FirstConnectionType() and NextConnectionType() are used to iterate through the available connection types. The functions return a BSTR,

which is the name of an available connection type. These returned types can be used with the SetConnectionType() function.

Return Value	
BSTR	The name of a supported connection device type.

BOOL FindConnectionType(LPCTSTR lpszConnectionType);

Returns TRUE if the passed in connection type exists on the computer.

Parameters	
LPCTSTR lpszConnectionType	The name of a connection device type.
Return Value	
BOOL	True if this device type is available.

short CountryCodes0;

Returns the number of country codes available.

Return Value	
Short	The number of country codes available.

BSTR FirstCountryCode0; and BSTR NextCountryCode0;

FirstCountryCode() and NextCountryCode() are used to iterate through the available country codes. The functions return a BSTR, which is the name of an available country code. These returned codes can be used with the SetCountryCode() function.

Return Value	
BSTR	The first or next country code string.

BOOL ReadObject(LPCTSTR lpszPassword);

Reads the object data from the Remote object file.

Parameters	
LPCTSTR lpszPassword	The object password.
Return Value	
BOOL	True if object is successfully read.

BOOL WriteObject(LPCTSTR lpszPassword);

Writes the object data out to the Remote object file.

Parameters	
LPCTSTR lpszPassword	The object password.
Return Value	
BOOL	True if object is successfully written.

CRemoteDataEx

The CRemoteDataEx object contains all the functionality of the CRemoteData object and the following Get Set methods:

```
BSTR GetPrivateKey(); //Returns the PrivateKey information

void SetPrivateKey(LPCTSTR lpszNewValue);

BSTR GetCertificationName(); //Returns the Certification
Name

void SetCertificationName(LPCTSTR lpszNewValue);

short GetEncryptionLevel(); //Returns the encryption level
value

void SetEncryptionLevel(short nNewValue);

BOOL GetDenyLowerEncrypt(); //Returns the DenyLowerEncryp
value

void SetDenyLowerEncrypt(BOOL bNewValue);
```

```
BSTR GetAutoDomain(); //Returns the AutoDomain value

void SetAutoDomain(LPCTSTR lpszNewValue);
```

C++ Sample Code

This sample C++ function creates a Remote object, sets its connection type to TCP/IP, sets the computer name to the TCP/IP address passed into the function, then launches the Remote object.

```
BOOL LaunchTCPRemote(LPCTSTR lpszAddress)
{
BOOL bReturn = FALSE;

CRemoteDataManager remoteDM;

CRemoteData remoteData;

// First, create the CRemoteDataManager
remoteDM.CreateDispatch( _T( "WINAWSVR.RemoteDataManager")
);
// Next, create CRemoteData and attach it
remoteData.AttachDispatch( remoteDM.CreateObject("Test", 0)
);

// Now, set the required properties
remoteData.SetConnectionType("TCP/IP");

remoteData.SetComputerName(lpszAddress);

// Save the object data
if (remoteData.WriteObject(0))
{
// And launch it
if (remoteData.Launch())
          bReturn = TRUE;
   }
   // Release the remote object.
   remoteData.ReleaseDispatch();

remoteDM.ReleaseDispatch( _T( "WINAWSVR.RemoteDataManager")
);

   return bReturn;
}
```

CHostDataManager

Methods

BSTR CurrentDirectory0;

Returns the full path name of the current directory in which pcAnywhere Host objects are stored.

Return Value	
BSTR	The full path name of the current pcAnywhere data directory.

BOOL ChangeDirectory(LPCTSTR lpszNewDirectory);

Changes the current directory in which pcAnywhere Host objects are stored.

Parameters	
LPCTSTR lpszNewDirectory	Name of an existing directory.
Return Value	
BOOL	True if successful.

BOOL FindFirst(LPCTSTR lpszPattern, BSTR FAR* pbstrFullQualName);

Finds the first pcAnywhere Host object file (*.BHF) in the current directory, based on the specified file name pattern.

Parameters	
LPCTSTR lpszPattern	File name pattern to filter object files ("*" finds all Host files in the current directory).
BSTR FAR * pbstrFullQualName	Return buffer for full path name of the remote object file matching the specified pattern.

Return Value	
BOOL	True if a Host object file matching the specified pattern is found. The full path name of the matching file is stored in pbstrFullQualName.

BOOL FindNext(BSTR FAR* pbstrFullQualName);

After FindFirst() has been successfully called to get the name of a Host object file in the current directory, FindNext() can be called to find the next file matching the pattern, if any.

Parameters	
BSTR FAR * pbstrFullQualName	Return buffer for full path name of the Host object file matching the pattern specified in the original call to FindFirst ( ) .
Return Value	
BOOL	True if another Host object file matching the pattern specified in the call to FindFirst() is found. The full path name of the matching file is stored in pbstrFullQualName.

LPDISPATCH RetrieveObject(LPCTSTR lpszFQName, short wAccessMode, LPCTSTR lpszPassword);

Retrieves a CHostData object by file name.

Parameters	
LPCTSTR lpszFQName	The fully qualified Host object file name to be loaded.
short wAccessMode	Specifies how this object is to be used. This is related to the password protection. The options include: 0 = Not specified 1 = View only 2 = View and Modify 3 = Execute

LPCTSTR lpszPassword	Object password. May be NULL.
Return Value	
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CHostData object.

LPDISPATCH RetrieveObjectEx(LPCTSTR lpszFQName, short wAccessMode, LPCTSTR lpszPassword);

Retrieves a CHostDataEx object by file name.

Parameters	
LPCTSTR lpszFQName	The fully qualified Host object file name to be loaded.
short wAccessMode	Specifies how this object is to be used. This is related to the password protection. The options include: 0 = Not specified 1 = View only 2 = View and Modify 3 = Execute
LPCTSTR lpszPassword	Object password. May be NULL.
Return Value	
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CHostDataEx object.

LPDISPATCH CreateObject(LPCTSTR lpszName);

Creates a CHostData object and returns an LPDISPATCH pointer to it.

Parameters	
LPCTSTR lpszFQName	The fully qualified Host object file name for new object.

Return Value	
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CHostData object.

LPDISPATCH CreateObjectEx(LPCTSTR lpszName);

Creates a CHostDataEx object and returns an LPDISPATCH pointer to it.

Parameters	
LPCTSTR lpszFQName	The fully qualified Host object file name for new object.
Return Value	
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CHostDataEx object.

BOOL DeleteObject(LPCTSTR lpszFQName, LPCTSTR lpszPassword);

Deletes a Host object file.

Parameters	
LPCTSTR lpszFQName	The fully qualified Host object file name of the object to be deleted.
LPCTSTR lpszPassword	Object password. May be NULL.
Return Value	
BOOL	True if object is deleted.

BOOL Launch(LPCTSTR lpszFQName);

Launches a Host object file. This opens the pcAnywhere Host terminal window.

Parameters	
LPCTSTR lpszFQName	The fully qualified Host object file of object to be launched.

Return Value	
BOOL	True if object is successfully launched.

CHostData

Use this object to modify Host object data.

Get and Set Methods

The following methods are used to get and set properties of the CHostData object.

```
BSTR GetComputerName();
void SetComputerName(LPCTSTR lpszNewValue);
```

The computer name is the name of the pcAnywhere remote computer to be called when the Host object is launched.

```
BSTR GetPhoneNumber();
void SetPhoneNumber(LPCTSTR lpszNewValue);
```

The phone number is the number to dial to establish a modem connection to a pcAnywhere remote computer.

```
BOOL GetUseDialingProperties();
void SetUseDialingProperties(BOOL bNewValue);
```

Indicates whether TAPI dialing properties should be used (location information) (TRUE), or the phone number string should be used exactly as it appears (FALSE).

```
BSTR GetAreaCode();
void SetAreaCode(LPCTSTR lpszNewValue);
```

If dialing properties are to be used, this is the area code of the number to be called.

```
BSTR GetCountryCode();
void SetCountryCode(LPCTSTR lpszNewValue);
```

If dialing properties are to be used, this is the country code of the number to be called.

```
short GetRedialCount();
void SetRedialCount(short nNewValue);
```

The number of times to retry dialing this number if the call fails.

```
short GetRedialDelay();
void SetRedialDelay(short nNewValue);
```

The time to wait (in seconds) between redial attempts.

```
BOOL GetLogSession();
void SetLogSession(BOOL bNewValue);
```

Controls whether sessions using this object are logged.

```
BOOL GetRecordSession();
void SetRecordSession(BOOL bNewValue);
```

Controls whether sessions using this object are recorded from the beginning.

```
BSTR GetRecordFile();
void SetRecordFile(LPCTSTR lpszNewValue);
```

The name of the record file for sessions using this object.

```
BOOL GetRunOnConnect();
void SetRunOnConnect(BOOL bNewValue);
```

Run one of the following procedures on connection (TRUE).

```
BSTR GetScriptFile();
void SetScriptFile(LPCTSTR lpszNewValue);
```

Script to run on connection (if RunOnConnect is TRUE).

```
BSTR GetAutoXferFile();
void SetAutoXferFile(LPCTSTR lpszNewValue);
```

AutoXfer commands to run on connection (if RunOnConnect is TRUE).

Detail Methods

These are methods used to get and set details for each of the connection types. When a Host object is assigned a connection type, the device details are set to valid default values.

COM device details

```
BOOL AssignConnection(LPCTSTR lpszNewValue);
```

Places the requested connection type on the Host object's list of assigned connection types, and makes it the current connection type when processing subsequent device-specific method calls. If the requested

connection type is already in the list of assigned connections, the list of assigned connections does not change; only the current connection type is changed to the requested type. It is normal to call the AssignConnection method on the same object multiple times in the course of getting and setting connection-specific values.

AssignConnection returns TRUE if the passed in connection type exists on the computer and is either successfully assigned or already assigned. It returns FALSE if either the requested connection type does not exist on the computer or the current assigned connection count is already at the maximum allowed level.

A pcAnywhere Host object can currently support up to two assigned connection types at any given time. The AssignConnection method returns FALSE if it detects an attempt to exceed this limit.

The available connection types include:

- COM1
- COM2
- COM3
- COM4
- SPX
- NetBIOS
- TCP/IP
- LPT1
- LPT2
- LPT3
- LPT4
- ISDN via CAPI 2.0
- Infrared
- DEFAULT TAPI

Also, the name of a TAPI device can be used as a connection type. “DEFAULT TAPI” uses the first TAPI device found in the system. To use a specific TAPI device, use FirstConnectionType() / NextConnectionType() to search for available devices.

```
BOOL UnassignConnection(LPCTSTR lpszNewValue);
```

Unassigns a connection type. After unassigning a connection type, the remaining assigned connection, if any, becomes the current connection type for subsequent device-specific method calls.

```
BSTR GetComParity();
void SetComParity(LPCTSTR lpszNewValue);
```

Values include:

- None
- Odd
- Even

- Mark
- Space

```
BSTR GetComFlowControl();void SetComFlowControl(LPCTSTR
lpszNewValue);
```

Values include:

- <None>
- XONXOFF
- RTS/CTS
- BOTH

```
BSTR GetComStartedBy();
void SetComStartedBy(LPCTSTR lpszNewValue);
```

Values include:

- Always connected
- Carrier detect (DCD)
- Clear to send (CTS)
- Data set ready (DSR)
- Ring indicator (RI)
- Receive 2 <CR>'s
- Modem response

```
BSTR GetComEndedBy();
void SetComEndedBy(LPCTSTR lpszNewValue);
```

Values include:

- Always connected
- Carrier detect (DCD)
- Clear to send (CTS)
- Data set ready (DSR)
- Ring indicator (RI)

```
long GetComSpeed();
void SetComSpeed(long nNewValue);
```

Values include:

- 110
- 300
- 600
- 1200

- 2400
- 4800
- 9600
- 19200
- 38400
- 57600
- 115200

Network (TCP/IP, SPX, Banyan) device details

```
BOOL GetGatewayUse();
void SetGatewayUse(BOOL bNewValue);
```

Connect through a pcAnywhere Gateway (TRUE).

```
BSTR GetGatewayName();
void SetGatewayName(LPCTSTR lpszNewValue);
```

Name of pcAnywhere Gateway to use.

```
BSTR GetGatewayClass();
void SetGatewayClass(LPCTSTR lpszNewValue);
```

Class of pcAnywhere Gateway to use.

```
BSTR GetGatewayParity();
void SetGatewayParity(LPCTSTR lpszNewValue);
```

Values are:

- <None>
- Odd
- Even
- Mark
- Space

NetBios device details

```
short GetLanaNumber();void SetLanaNumber(short nNewValue);
```

The LANA (LAN Adapter) number to use for this connection.

NASI/NCSI device details

```
BSTR GetNasiUserName();
void SetNasiUserName(LPCTSTR lpszNewValue);
```

User name for NASI server.

```
BSTR GetNasiPassword();
void SetNasiPassword(LPCTSTR lpszNewValue);
```

User password for NASI server.

```
BSTR GetNasiSessionName();
void SetNasiSessionName(LPCTSTR lpszNewValue);
```

NASI session name.

```
BOOL GetNasiSessionNameAvailable();
void SetNasiSessionNameAvailable(BOOL bNewValue);
BOOL NasiServer();
BSTR GetNasiServerName();
void SetNasiServerName(LPCTSTR lpszNewValue);
```

Specify the NASI server to use.

```
BOOL NasiService();
BSTR GetNasiServiceName();
void SetNasiServiceName(LPCTSTR lpszNewValue);
```

Specify the NASI Service to use.

```
BOOL NasiPort();
BSTR GetNasiPortName();
void SetNasiPortName(LPCTSTR lpszNewValue);
```

Specify the NASI Port to use.

```
BOOL GetNasiSelectOnConnect();
void SetNasiSelectOnConnect(BOOL bNewValue);
```

ISDN via CAPI 2.0 device details

```
BOOL GetCapiChannelBonding();
void SetCapiChannelBonding(BOOL bNewValue);
```

Use Channel Bonding (uses 2 ISDN channels for one connection) (TRUE).

```
BSTR GetCapiExtensions();
void SetCapiExtensions(LPCTSTR lpszNewValue);
```

Methods

The following are the normal methods of the object. They are not used to get and set properties.

short ConnectionTypes0;

Returns the number of connection types available.

Return Value	
Short	The number of connection types actually available on this system.

BSTR FirstConnectionType0; and BSTR NextConnectionType0;

FirstConnectionType() and NextConnectionType() are used to iterate through the available connection types. The functions return a `BSTR`, which is the name of an available connection type. These returned types can be used with the SetConnectionType() function.

Return Value	
BSTR	The name of a supported connection device type.

BOOL FindConnectionType(LPCTSTR lpszConnectionType);

Returns TRUE if the passed in connection type exists on the computer.

Parameters	
LPCTSTR lpszConnectionType	The name of a connection device type.
Return Value	
BOOL	True if this device type is available.

short MaxAssignedConnections0

Returns the maximum allowed number of assigned connections (currently two).

Return Value	
Short	The maximum allowed number of assigned connections.

short AssignedConnections0

Returns the number of assigned connection types.

Return Value	
Short	The number of connection types currently assigned on this system.

BSTR FirstAssignedConnection0; and BSTR NextAssignedConnection 0;

FirstAssignedConnection() and NextAssignedConnection() are used to iterate through the list of assigned connections. The functions return a BSTR, which is the name of an assigned connection type. These returned types can be used with the AssignConnection() function.

Return Value	
BSTR	The name of a supported connection device type.

BOOL FindAssignedConnection (LPCTSTR lpszConnectionType);

Returns TRUE if the passed in connection type is currently assigned on the computer.

Parameters	
LPCTSTR lpszConnectionType	The name of a connection device type.
Return Value	
BOOL	True if this device type is currently assigned.

short CountryCodes0;

Returns the number of country codes available.

Return Value	
Short	The number of country codes available.

BSTR FirstCountryCode0; and BSTR NextCountryCode0;

FirstCountryCode() and NextCountryCode() are used to iterate through the available country codes. The functions return a BSTR, which is the name of an available country code. These returned codes can be used with the SetCountryCode() function.

Return Value	
BSTR	The first or next country code string.

BOOL ReadObject(LPCTSTR lpszPassword);

Reads the object data from the Host object file.

Parameters	
LPCTSTR lpszPassword	The object password.
Return Value	
BOOL	True if object is successfully read.

BOOL WriteObject(LPCTSTR lpszPassword);

Writes the object data out to the Host object file.

Parameters	
LPCTSTR lpszPassword	The object password.
Return Value	
BOOL	True if object is successfully written.

CHostDataEx

The CHostDataEx contains the same functionality as the CHostData, with the added Get Set methods below:

Additional functionality

BOOL GetReadProtection();

void SetReadProtection(BOOL bNewValue);

BOOL GetWriteProtection();

void SetWriteProtection(BOOL bNewValue);

BSTR GetPassword(); //Returns “NOT IMPLEMENTED”

void SetPassword(LPCTSTR lpszNewValue);

BSTR GetCallersPath();

void SetCallersPath(LPCTSTR lpszNewValue);

BOOL GetConfirmConnect();

void SetConfirmConnect(BOOL bNewValue);

short GetConfirmTimeout();

void SetConfirmTimeout(short nNewValue);

BOOL GetConfirmDeny();

void SetConfirmDeny(BOOL bNewValue);

BOOL GetPwCaseSensitive();

void SetPwCaseSensitive(BOOL bNewValue);

short GetPwAttempts();

void SetPwAttempts(short nNewValue);

short GetPwTimeout();

void SetPwTimeout(short nNewValue);

short GetActiveKbds();

```
void SetActiveKbds(short nNewValue); //Sets ActiveKbds
short GetInactiveTimeout();
void SetInactiveTimeout(short nNewValue);
short GetCryptReqLevel();
void SetCryptReqLevel(short nNewValue);
BOOL GetCryptRefuseLower();
void SetCryptRefuseLower(BOOL bNewValue);
short GetAuthenticationType();
void SetAuthenticationType(short nNewValue);
BOOL GetLockSystemWhileWait();
void SetLockSystemWhileWait(BOOL bNewValue);
BOOL GetMinimizeOnLaunch();
void SetMinimizeOnLaunch(BOOL bNewValue);
BOOL GetRunAsService();
void SetRunAsService(BOOL bNewValue);
short GetConnLostWait();
void SetConnLostWait(short nNewValue);
BOOL GetConnLostHostOpts();
void SetConnLostHostOpts(BOOL bNewValue);
BOOL GetEnableConnLostSecurity();
void SetEnableConnLostSecurity(BOOL bNewValue);
short GetConnLostSecurity();
void SetConnLostSecurity(short nNewValue);
short GetCallbkDelay();
void SetCallbkDelay(short nNewValue);
```

```
BOOL GetEndSessHostOpts();
void SetEndSessHostOpts(BOOL bNewValue);
BOOL GetEnableEndSessSecurity();
void SetEnableEndSessSecurity(BOOL bNewValue);
short GetEndSessSecurity();
void SetEndSessSecurity(short nNewValue);
BSTR GetCryptPrivateKey();
void SetCryptPrivateKey(LPCTSTR lpszNewValue);
BSTR GetCryptCommonName();
void SetCryptCommonName(LPCTSTR lpszNewValue);
BOOL GetBlankHost();
void SetBlankHost(BOOL bNewValue);
BOOL GetAllowRemoteMouse();
void SetAllowRemoteMouse(BOOL bNewValue);
short GetRebootOnDisconnect();
void SetRebootOnDisconnect(short nNewValue);
BOOL GetPasswordAfterDisc();
void SetPasswordAfterDisc(BOOL bNewValue);
BOOL GetLogFailures();
void SetLogFailures(BOOL bNewValue);
BOOL GetAllowDriveSecurity();
void SetAllowDriveSecurity(BOOL bNewValue);
BOOL GetExecuteProtection();
void SetExecuteProtection(BOOL bNewValue);
```

C++ Sample Code

This sample C++ function creates a Host object, sets its connection type to TCP/IP, sets the computer name to the TCP/IP address passed into the function, then launches the Host object.

```
BOOL LaunchTCPHost(LPCTSTR lpszAddress)
{
BOOL bReturn = FALSE;

CHostDataManager hostDM;
CHostData hostData;

// First, create the CHostDataManager
hostDM.CreateDispatch( _T( "WINAWSVR.BeHostDataManager") );

// Next, create CRemoteData and attach it
hostData.AttachDispatch(hostDM.CreateObject("Test", 0) );

// Now, set the required properties
hostData.SetConnectionType("TCP/IP");
hostData.SetComputerName(lpszAddress);

// Save the object data
if (hostData.WriteObject(0))
{
// And launch it
if (hostData.Launch())
   bReturn = TRUE;
   }

   // Release the Host object.
   hostData.ReleaseDispatch();

   return (bReturn);
}
```

AWREM32 functions

boolean awConnect(BSTR FileName);

Creates the connection to the Host computer.

Parameters	
Name as string	The fully qualified .chf file name that contains information about the Host computer.
Return Value	
Boolean	Executes the command.

boolean awDisconnect0;

Disconnects the Host computer.

Return Value	
Boolean	After calling this function, the calling program must delete the object (C++ - delete IAwrem32X*, VB – set ObjectName = Nothing;).

boolean FileXferFromHost(BSTR HostFile, BSTR RemoteFile);

Copies a file from the Host computer to the Remote computer. The parameters can contain wildcards.

Parameters	
HostFile as string	Contains the fully qualified path and file name to be copied.
RemoteFile as string	Contains the fully qualified path and file name. The HostFile and RemoteFile strings do not have to be identical.
Return Value	
Boolean	True if command executed.

boolean FileXferToHost(BSTR HostFile, BSTR RemoteFile);

Copies a file from the Remote computer to the Host computer. The parameters can contain wildcards.

Parameters	
HostFile as string	Contains the fully qualified destination path and file name.
RemoteFile as string	Contains the fully qualified path and file name to be copied. The HostFile and RemoteFile strings do not have to be identical.
Return Value	
Boolean	True if command executed.

boolean CreateFolderOnHost(BSTR FolderName);

Creates a new folder on the Host computer. It creates a temporary folder on the Remote computer, then copies that folder to the Host.

Parameters	
FolderName as string	Contains the drive and path to create the folder on the host computer.
Return Value	
Boolean	True if command executed.

boolean ExecuteHostFile(BSTR FileName);

Executes an existing file on the Host computer. This function only executes batch, command, and executable files. It does not execute files that are associated with executables. For example, it does not open Microsoft Word if you execute a .doc file.

Parameters	
FileName as string	Contains the fully qualified path to the file on the Host computer.
Return Value	
Boolean	True if command executed.

BSTR GetError0;

Returns the last error as a string

Return Value	
String	Returns the last error generated in AWREM32.

short ConnectionStatus0;

Returns the current status of your connection to the Host computer.

Return Value	
Short	-1 = Lost Connection 0 = No Connection 1 = Session Connected

C H A P T E R 3

Visual Basic Object Definitions

Some functions, including Gateways, are not available in pcAnywhere 10.0, but are included here for use with previous versions.

For functions involving passwords, password values can be set, but not retrieved. This is for security purposes.

CRemoteDataManager

Methods

CurrentDirectory()

Returns the full path name of the current directory where pcAnywhere objects are stored.

Return Value	
String	The full path name of the current pcAnywhere data directory.

ChangeDirectory(NewDirectory)

Changes the current directory where pcAnywhere objects are stored.

Parameters	
NewDirectory	Name of an existing directory.

Return Value	
Boolean	True if successful.

FindFirst(Pattern, Name string)

Finds the first pcAnywhere Remote object file (*.CHF) in the current directory, based on the specified file name pattern.

Parameters	
Pattern as string	File name pattern to filter object files (“*”finds all files in the current directory).
Name as string	Return buffer for full path name of the remote object file matching the specified pattern.
Return Value	
Boolean	True if a remote object file matching the specified pattern is found. The full path name of the matching file is stored in Name.

FindNext(Name)

After FindFirst() has been successfully called to get the name of a remote object file in the current directory, FindNext() can be called to find the next file matching the pattern, if any.

Parameters	
Name as string	Return buffer for full path name of the remote object file matching the pattern specified in the original call to FindFirst().
Return Value	
Boolean	True if another remote object file matching the pattern specified in the call to FindFirst() is found. The full path name of the matching file is stored in Name.

RetrieveObject(Name, AccessMode, Password)

Retrieves a CRemoteData object by file name.

Parameters	
Name as string	The fully qualified remote object file name to be loaded.
AccessMode as integer	Specifies how this object is to be used. This is related to the password protection. The options include: 0 = Not specified 1 = View only 2 = View and Modify 3 = Execute
Password as string	Object password. May be NULL.
Return Value	
Object	CRemoteData object from the specified file.

RetrieveObjectEx(Name, AccessMode, Password)

Retrieves a CRemoteDataEx object by file name.

Parameters	
Name as string	The fully qualified remote object file name to be loaded.
AccessMode as integer	Specifies how this object is to be used. This is related to the password protection. The options include: 0 = Not specified 1 = View only 2 = View and Modify 3 = Execute
Password as string	Object password. May be NULL.

Return Value	
Object	CRemoteDataEx object from the specified file.

CreateObject(Name)

Creates a CRemoteData object and returns an LPDISPATCH pointer to it.

Parameters	
Name as string	The fully qualified remote object file name for new object.
Return Value	
Object	CRemoteData

CreateObjectEx(Name)

Creates a CRemoteDataEx object and returns an LPDISPATCH pointer to it.

Parameters	
Name as string	The fully qualified remote object file name for new object.
Return Value	
Object	CRemoteDataEx

DeleteObject(Name, Password)

Deletes a remote object file.

Parameters	
Name as string	The fully qualified remote object file name of the object to be deleted.
Password as string	Object password.
Return Value	
Boolean	True if object is deleted.

CRemoteData

Properties

- ComputerName as string
- PhoneNumber as string
- UseDialingProperties as boolean
- AreaCode as string
- CountryCode as string
- RedialCount as integer
- RedialDelay as integer
- AutoLoginName as string
- AutoLoginPassword as string
- Password as string
- ExecuteProtection as boolean
- ReadProtection as boolean
- WriteProtection as boolean
- LogSession as boolean
- RecordFile as string
- RecordSession as boolean
- RunOnConnect as boolean
- AutoXferFile as string
- ConnectionType as string

Connection types available on a computer can be obtained with the FirstConnectionType() and NextConnectionType() functions. The connection types available include:

- COM1
- COM2
- COM3
- COM4
- SPX
- NetBIOS
- TCP/IP
- LPT1
- LPT2
- LPT3
- LPT4
- ISDN via CAPI 2.0
- Infrared
- DEFAULT TAPI

Also, the name of a TAPI device can be used as a connection type. DEFAULT TAPI uses the first TAPI device found in the system. To use a specific TAPI device, you must use FirstConnectionType() / NextConnectionType() to search for available devices.

Detail Properties

These are properties specific to the different connection types. When a remote object is assigned a connection type, the device details are set to valid default values.

COM device properties

ComParity as string

Values include:

- <None>
- Odd
- Even
- Mark
- Space

ComFlowControl as string

Values include:

- <None>
- XONXOFF
- RTS/CTS
- BOTH

ComStartedBy as string

Values include:

- Always connected
- Carrier detect (DCD)
- Clear to send (CTS)
- Data set ready (DSR)
- Ring indicator (RI)

- Receive 2 <CR>'s
- Modem response

ComEndedBy as string

Values include:

- Always connected
- Carrier detect (DCD)
- Clear to send (CTS)Data set ready (DSR)
- Ring indicator (RI)

ComSpeed as long

Values include:

- 110
- 300
- 600
- 1200
- 2400
- 4800
- 9600
- 19200
- 38400
- 57600
- 115200

Network (TCP/IP, SPX, NetBios) device properties

- GatewayUse as boolean
- GatewayName as string
- GatewayClass as string
- GatewayParity as string

NetBios

LanaNumber as integer

ISDN via CAPI 2.0 device properties

- CapiChannelBonding as boolean
- CapiExtensions as string

Methods

The following are the normal methods of the object.

ConnectionTypes()

Returns the number of connection types available.

Return Value	
Integer	The number of connection types actually available on this system.

FirstConnectionType() and NextConnectionType()

FirstConnectionType() and NextConnectionType() are used to iterate through the available connection types. The functions return a string, which is the name of an available connection type. These returned types can be used with the SetConnectionType() function.

Return Value	
String	The name of a supported connection device type.

FindConnectionType(ConnectionType)

Returns TRUE if the passed in connection type exists on the machine.

Parameters	
ConnectionType as string	The name of a connection device type.
Return Value	
Boolean	True if this device type is available.

CountryCodes()

Returns the number of country codes available.

Return Value	
Integer	The number of country codes available.

FirstCountryCode() and NextCountryCode()

FirstCountryCode() and NextCountryCode() are used to iterate through the available country codes. The functions return a string, which is the name of an available country code. These returned codes can be used with the SetCountryCode() function.

Return Value	
String	The first or next country code string.

ReadObject(Password)

Reads the object data from the remote object file.

Parameters	
Password as string	The object password.
Return Value	
Boolean	True if object is successfully read.

WriteObject(Password)

Writes the object data out to the Remote object file.

Parameters	
Password as string	The object password.
Return Value	
Boolean	True if object is successfully written.

CRemoteDataEx

CRemoteDataEx contains the same functionality as CRemoteData, with the addition of the following functionality:

```
PrivateKey as string          //Container name of Private Key

CertificationName as string   //The common name associated with the private key
```

```
EncryptionLevel as byte          //The level of encryption, -1 = None, 0 =
                                 pcAnywhere, 1 = Symmetric, 2 = Public key

DenyLowerEncrypt as              //Allow or Deny communication below that of the
boolean                          EncryptionLevel value

AutoDomain as string             //Combined with autologin and autopassword for the
                                 three variables required for automatic login
```

Visual Basic sample code

This sample Visual Basic sample code retrieves a Remote Data Object and modifies its properties.

```
Private Sub Command1_Click()
Dim RemoteDataManager as Object
Dim RemoteData as Object
Dim s as string

'Create CRemoteDataManager object
Set RemoteDataManager =
CreateObject(WINAWSVR.REMOTEDATAMANAGER)

'display and change current directory
s = RemoteDataManager.CurrentDirectory()
MsgBox ( s )
RemoteDataManager.ChangeDirectory
("C:\dev\bin.w32\data")
s = RemoteDataManager.CurrentDirectory()
MsgBox ( s )

'retrieve remote data object
Set RemoteData =
RemoteDataManager.RetrieveObjectEx("pod.CHF", 2, 0)

'display some properties
s = RemoteData.AreaCode()
MsgBox (s)
s = RemoteData.PhoneNumber()
MsgBox (s)

'set some properties
RemoteData.AreaCode = "212"
RemoteData.PhoneNumber = "555-5555"

'write object to disk
RemoteData.WriteObject (0)
End Sub
```

Use the FindFirst and FindNext methods to display Remote file in a directory.

```
Private Sub Command5_Click()

Dim RemoteDataManager as Object
Dim RemoteData as Object
Dim s as string

Set RemoteDataManager =
CreateObject("WINAWSVR.REMOTEDATAMANAGER")
RemoteDataManager.ChangeDirectory
("C:\dev\bin.w32\data")
RemoteDataManager.FindFirst "*", s
MsgBox (s)
RemoteDataManager.FindNext s
MsgBox (s)

End Sub
```

Create a Remote object, set connection for TCP/IP, computer name "Host1," and then launch it.

```
Private Sub Command6_Click()
Dim RemoteDataManager as Object
Dim RemoteData as Object
Dim s as string

Set RemoteDataManager =
CreateObject("WINAWSVR.REMOTEDATAMANAGER")
MsgBox (RemoteDataManager.CurrentDirectory())

RemoteDataManager.ChangeDirectory
("C:\dev\bin.w32\data")
MsgBox (RemoteDataManager.CurrentDirectory())
Set RemoteData = RemoteDataManager.CreateObject("test")
RemoteData.ConnectionType = "TCP/IP"
RemoteData.ComputerName = "Host1"
s = RemoteData.ConnectionType
MsgBox (s)
s = RemoteData.ComputerName
MsgBox (s)
RemoteData.WriteObject (0)
End Sub
```

CHostDataManager

Methods

CurrentDirectory0

Returns the full path name of the current directory where pcAnywhere Host objects are stored.

Return Value	
String	The full path name of the current pcAnywhere data directory.

FindNext(Name)

After FindFirst() has been successfully called to get the name of a Host object file in the current directory, FindNext() can be called to find the next file matching the pattern, if any.

Parameters	
Name as string	Return buffer for full path name of the Host object file matching the pattern specified in the original call to FindFirst().
Return Value	
Boolean	True if another Host object file matching the pattern specified in the call to FindFirst() is found. The full path name of the matching file is stored in Name.

RetrieveObject(Name, AccessMode, Password)

Retrieves a CHostData object by file name.

Parameters	
Name as string	The fully qualified Host object file name to be loaded.
AccessMode as integer	Specifies how this object is to be used. This is related to the password protection. The options include: 0 = Not specified 1 = View only 2 = View and Modify 3 = Execute
Password as string	Object password. May be NULL.
Return Value	
Object	CRemoteData object from the specified file.

RetrieveObjectEx(Name, AccessMode, Password)

Retrieves a CHostDataEx object by file name.

Parameters	
Name as string	The fully qualified Host object file name to be loaded.
AccessMode as integer	Specifies how this object is to be used. This is related to the password protection. The options include: 0 = Not specified 1 = View only 2 = View and Modify 3 = Execute
Password as string	Object password. May be NULL.
Return Value	
Object	CRemoteDataEx object from the specified file.

CreateObject(Name)

Creates a CHostData object and returns an LPDISPATCH pointer to it.

Parameters	
Name as string	The fully qualified Host object file name for new object.
Return Value	
Object	CHostData

CreateObjectEx(Name)

Creates a CHostDataEx object and returns an LPDISPATCH pointer to it.

Parameters	
Name as string	The fully qualified Host object file name for new object.

Return Value	
Object	CHostDataEx

DeleteObject(Name, Password)

Deletes a Host object file.

Parameters	
Name as string	The fully qualified Host object file name of the object to be deleted.
Password as string	Object password.
Return Value	
Boolean	True if object is deleted.

Launch(Name)

Launches a Host object file. This opens the pcAnywhere Host terminal window.

Parameters	
Name as string	The fully qualified Host object file of object to be launched.
Return Value	
Boolean	True if object is successfully launched.

CHostData

Properties

- ComputerName as string
- PhoneNumber as string
- UseDialingProperties as boolean
- AreaCode as string
- CountryCode as string

- RedialCount as integer
- RedialDelay as integer
- LogSession as boolean
- RecordFile as string
- RecordSession as boolean
- RunOnConnect as boolean
- ScriptFile as string
- AutoXferFile as string

Detail Properties

These are properties specific to the different connection types. When a remote object is assigned a connection type, the device details are set to valid default values.

COM device properties

ComParity as string

Values include:

- <None>
- Odd
- Even
- Mark
- Space

ComFlowControl as string

Values include:

- <None>
- XONXOFF
- RTS/CTS
- BOTH

ComStartedBy as string

Values include:

- Always connected
- Carrier detect (DCD)
- Clear to send (CTS)
- Data set ready (DSR)
- Ring indicator (RI)
- Receive 2 <CR>'s
- Modem response

ComEndedBy as string

Values include:

- Always connected
- Carrier detect (DCD)
- Clear to send (CTS)Data set ready (DSR)
- Ring indicator (RI)

ComSpeed as long

Values include:

- 110
- 300
- 600
- 1200
- 2400
- 4800
- 9600
- 19200
- 38400
- 57600
- 115200

Network (TCP/IP, SPX, Banyan) device properties

- GatewayUse as boolean
- GatewayName as string
- GatewayClass as string
- GatewayParity as string

NetBios

LanaNumber as integer

ISDN via CAPI 2.0 device properties

- CapiChannelBonding as boolean
- CapiExtensions as string

Methods

The following are the normal methods of the object.

ConnectionTypes0

Returns the number of connection types available.

Return Value	
Integer	The number of connection types actually available on this system.

The connection types available include:

- COM1
- COM2
- COM3
- COM4
- SPX
- NetBIOS
- TCP/IP
- LPT1
- LPT2
- LPT3
- LPT4
- ISDN via CAPI 2.0
- Infrared
- DEFAULT TAPI

Also, the name of a TAPI device can be used as a connection type. “DEFAULT TAPI” uses the first TAPI device found in the system. To use a specific TAPI device, use FirstConnectionType() / NextConnectionType() to search for available devices.

FirstConnectionType0 and NextConnectionType0

FirstConnectionType() and NextConnectionType() are used to iterate through the available connection types. The functions return a string, which is the name of an available connection type. These returned types can be used with the AssignConnection() function.

Return Value	
String	The name of a supported connection device type.

FindConnectionType(ConnectionType)

Returns TRUE if the passed in connection type exists on the computer.

Parameters	
ConnectionType as string	The name of a connection device type.
Return Value	
Boolean	True if this device type is available.

MaxAssignedConnections0

Returns the maximum number of connection types that can be assigned at the same time.

Return Value	
Integer	The maximum number of connection type assignments.

AssignedConnections0

Returns the number of currently assigned connection types.

Return Value	
Integer	The number of assigned connection types.

FirstAssignedConnection0 and NextAssignedConnection0

FirstAssignedConnection() and NextAssignedConnection() are used to iterate through the currently assigned connection types. The functions return a string, which is the name of an available connection type. These returned types can be used with the AssignConnection() function.

Return Value	
String	The name of a supported connection device type.

FindAssignedConnection(ConnectionType)

Returns TRUE if the passed in connection type is currently assigned on the computer.

Parameters	
ConnectionType as string	The name of a connection device type.
Return Value	
Boolean	True if this device type is currently assigned.

AssignConnection(ConnectionType)

Places the requested connection type on the Host object's list of assigned connection types, and makes it the current connection type when processing subsequent device-specific method calls. If the requested connection type is already in the list of assigned connections, the list of assigned connections does not change; only the current connection type is changed to the requested type. It is normal to call the AssignConnection method on the same object multiple times in the course of getting and setting connection-specific values.

AssignConnection returns TRUE if the passed in connection type exists on the computer and is either successfully assigned or already assigned. It returns FALSE if either the requested connection type does not exist on the computer or the current assigned connection count is already at the maximum allowed level.

A pcAnywhere Host object can currently support up to two assigned connection types at any given time. The AssignConnection method returns FALSE if it detects an attempt to exceed this limit.

Parameters	
ConnectionType as string	The name of a connection device type to be assigned.
Return Value	
Boolean	True if this device type is available and the maximum allowed assigned connection count has not already been reached.

UnassignConnection(ConnectionType)

Returns TRUE if the passed in connection type is successfully removed from the list of assigned connection types.

Parameters	
ConnectionType as string	The name of a connection device type to be unassigned.
Return Value	
Boolean	True if this device type is successfully unassigned.

CountryCodes0

Returns the number of country codes available.

Return Value	
Integer	The number of country codes available.

FirstCountryCode() and NextCountryCode()

FirstCountryCode() and NextCountryCode() are used to iterate through the available country codes. The functions return a string, which is the name of an available country code. These returned codes can be used with the SetCountryCode() function.

Return Value	
String	The first or next country code string.

ReadObject(Password)

Reads the object data from the remote object file.

Parameters	
Password as string	The object password.
Return Value	
Boolean	True if object is successfully read.

WriteObject(Password)

Writes the object data out to the Remote object file.

Parameters	
Password as string	The object password.
Return Value	
Boolean	True if object is successfully written.

CHostDataEx

CHostDataEx contains the same functionality as CHostData, with the added functionality shown below:

```
ReadProtection as boolean       // Require password to read this file

WriteProtection as boolean      // Require password to write to this file

Password as string              //Password to protect object
```

```
CallersPath as string                  //Path to caller objects
ConfirmConnect as boolean              //Allow or deny Host operator to confirm
                                       connection
ConfirmTimeout as byte                 //Time for Host operator to confirm connection
                                       or disconnect
ConfirmDeny as boolean                 //Allow or deny Host operator to confirm Deny
                                       connection
PwCaseSensitive as boolean             //Is password case sensitive
PwAttempts as byte                     //Number of times login attempts allowed
                                       before disconnect
PwTimeout as byte                      //Time allowed to complete successful login
                                       before disconnect
ActiveKbds as byte                     // 0 = Host and Remote, 1 = Host, 2 = Remote
InactiveTimeout as byte                //Time limit for activity between Remote and
                                       Host before disconnect
CryptReqLevel as byte                  //Preferred level of encryption, -1 = None, 0 =
                                       pcAnywhere, 1 = Symmetric, 2 = Public key
CryptRefuseLower as boolean            //Preferred level is minimum acceptable level of
                                       encryption
AuthenticationType as byte             //Returns index of Authentication Type
                                       comboBox
LockSystemWhileWait as boolean         //Settings – Host StartUp option
MinimizeOnLaunch as boolean            //Settings – Host StartUp option
RunAsService as boolean                //Settings – Host StartUp option
ConnLostWait as byte                   //Settings – Abnormal End of Session Option
                                       Time to wait before reconnect allowed
ConnLostHostOpts as boolean            //Wait or Cancel Host
EnableConnLostSecurity as              //Enable ConnLostSecurity
boolean
```

ConnLostSecurity as byte	// 1 = Log off user, 2 = Restart host computer, 3 = Lock NT
CallbkDelay as byte	//Modem callback delay in seconds (0-9999)
EndSessHostOpts as boolean	//Wait or Cancel host
EnableEndSessSecurity;	// Enable EndSessSecurity
EndSessSecurity as byte	// 1 = Log off user, 2 = Restart host computer, 3 = Lock NT
CryptPrivateKey as string	//Container name of Private Key
CryptCommonName as string	//Certificate Common Name
BlankHost as boolean	//Black PC screen after Connection
AllowRemoteMouse as boolean	//T F
RebootOnDisconnect as byte	//Reboot on disconnect
PasswordAfterDisc as boolean	//Require revalidation of password after disconnect
LogFailures as boolean	//Include Login failure attempts in log
AllowDriveSecurity as boolean	//
ExecuteProtection as boolean	//Require password to execute file
UseDirectoryServices as boolean	
DirectoryServiceEntry as string	

Visual Basic sample code

Retrieve a Host Data Object and modify its properties.

```
Private Sub Command1_Click()
Dim HostDataManager as Object
Dim HostData as Object
Dim s as string

'Create CHostDataManager object
Set HostDataManager =
CreateObject(WINAWSVR.BEHOSTDATAMANAGER)

'display and change current directory
s = HostDataManager.CurrentDirectory()
MsgBox ( s )
HostDataManager.ChangeDirectory ("C:\dev\bin.w32\data")
s = HostDataManager.CurrentDirectory()
MsgBox ( s )

'retrieve remote data object
Set HostData = HostDataManager.RetrieveObject("pod.BHF",
2, 0)

'display some properties
s = HostData.AreaCode()
MsgBox (s)
s = HostData.PhoneNumber()
MsgBox (s)
'set some properties
RemoteData. HostData = "212"
RemoteData. HostData = "555-5555"

'write object to disk
HostData.WriteObject (0)
End Sub
```

Use the FindFirst and FindNext methods to display Host file in a directory.

```
Private Sub Command5_Click()

Dim HostDataManager as Object
Dim HostData as Object
Dim s as string

Set HostDataManager =
CreateObject("WINAWSVR.BEHOSTDATAMANAGER")
HostDataManager.ChangeDirectory ("C:\dev\bin.w32\data")
HostDataManager.FindFirst "*", s
MsgBox (s)
HostDataManager.FindNext s
MsgBox (s)

End Sub
```

AWREM32 functions

awConnect(FileName)

Creates the connection to the Host computer.

Parameters	
Name as string	The fully qualified .chf file name that contains information about the Host computer.
Return Value	
Boolean	Executes the command.

awDisconnect0

Disconnects the Host computer.

Return Value	
Boolean	After calling this function, the calling program must delete the object (C++ - delete IAwrem32X*, VB – set ObjectName = Nothing;).

FileXferFromHost(HostFile, RemoteFile)

Copies a file from the Host computer to the Remote computer. The parameters can contain wildcards.

Parameters	
HostFile as string	Contains the fully qualified path and file name to be copied.
RemoteFile as string	Contains the fully qualified path and file name. The HostFile and RemoteFile strings do not have to be identical.
Return Value	
Boolean	True if command executed.

FileXferToHost(HostFile, RemoteFile)

Copies a file from the Remote computer to the Host computer. The parameters can contain wildcards.

Parameters	
HostFile as string	Contains the fully qualified destination path and file name.
RemoteFile as string	Contains the fully qualified path and file name to be copied. The HostFile and RemoteFile strings do not have to be identical.
Return Value	
Boolean	True if command executed.

CreateFolderOnHost(FolderName)

Creates a new folder on the Host computer. It creates a temporary folder on the Remote computer, then copies that folder to the Host.

Parameters	
FolderName as string	Contains the drive and path to create the folder on the host computer.

Return Value	
Boolean	True if command executed.

ExecuteHostFile(FileName)

Executes an existing file on the Host computer. This function only executes batch, command, and executable files. It does not execute files that are associated with executables. For example, it does not open Microsoft Word if you execute a .doc file.

Parameters	
FileName as string	Contains the fully qualified path to the file on the Host computer.
Return Value	
Boolean	True if command executed.

GetError0

Returns the last error as a string

Return Value	
String	Returns the last error generated in AWREM32.

ConnectionStatus0

Returns the current status of your connection to the Host computer.

Return Value	
Short	-1 = Lost Connection 0 = No Connection 1 = Session Connected