

Symantec pcAnywhere™ OLE Automation Guide

Symantec pcAnywhere™ OLE Automation Guide

The software described in this book is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Documentation version 11.0

Copyright Notice

Copyright © 2003 Symantec Corporation.

All Rights Reserved.

Any technical documentation that is made available by Symantec Corporation is the copyrighted work of Symantec Corporation and is owned by Symantec Corporation.

NO WARRANTY. The technical documentation is being delivered to you AS-IS, and Symantec Corporation makes no warranty as to its accuracy or use. Any use of the technical documentation or the information contained therein is at the risk of the user. Documentation may include technical or other inaccuracies or typographical errors. Symantec reserves the right to make changes without prior notice.

No part of this publication may be copied without the express written permission of Symantec Corporation, 20330 Stevens Creek Blvd., Cupertino, CA 95014.

Trademarks

Symantec, the Symantec logo, and pcAnywhere are U.S. registered trademarks of Symantec Corporation.

Other brands and product names mentioned in this manual may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Contents

Chapter 1 Using OLE Automation with Symantec pcAnywhere

About OLE Automation	8
About the pcAnywhere Automation Server	8
What you can do with the pcAnywhere Automation Server	9
Before you start	9
Automatically registering the remote engine	9
Manually registering the remote engine	10
Accessing the pcAnywhere Automation Server	10
Accessing the pcAnywhere Automation Server with Visual Basic	10
Accessing the pcAnywhere Automation Server with Visual C++	11
Launching host and remote OLE objects	13
Where to find more information	14

Chapter 2 Visual Basic object definitions

About Visual Basic objects	16
CRemoteDataManager methods	17
CurrentDirectory()	17
ChangeDirectory(NewDirectory)	17
FindFirst(Pattern, Name string)	18
FindNext(Name)	18
RetrieveObject(Name, AccessMode, Password)	19
RetrieveObjectEx(Name, AccessMode, Password)	20
CreateObject(Name)	20
CreateObjectEx(Name)	21
DeleteObject(Name, Password)	21
CRemoteData properties	22
Connection type properties	24
Dialing properties	26
COM device properties	27
NetBIOS device properties	30
ISDN via CAPI 2.0 device properties	31
Network (TCP/IP, SPX) device properties for Gateways	31
CRemoteDataEx object	32
Visual Basic sample code for remote functionality	33

CHostDataManager methods	35
CurrentDirectory()	35
FindFirst(Pattern, Name string)	36
FindNext(Name)	36
RetrieveObject(Name, AccessMode, Password)	37
RetrieveObjectEx(Name, AccessMode, Password)	38
CreateObject(Name)	38
CreateObjectEx(Name)	39
DeleteObject(Name, Password)	39
Launch(Name)	40
CHostData properties	40
Connection type properties	42
AssignConnection(ConnectionType) method	44
UnassignConnection(ConnectionType) method	45
Dialing properties	46
COM device properties	47
NetBIOS device properties	50
ISDN via CAPI 2.0 device properties	51
Network (TCP/IP, SPX) device properties for Gateways	51
CHostDataEx object	52
Visual Basic sample code for host functionality	60
Awrem32 functions	61
awConnect(FileName)	61
awDisconnect()	62
FileXferFromHost(HostFile, RemoteFile)	62
FileXferToHost(HostFile, RemoteFile)	63
CreateFolderOnHost(FolderName)	63
ExecuteHostFile(FileName)	64
GetError()	64
ConnectionStatus()	64

Chapter 3 Visual C++ object definitions

About Visual C++ objects	66
CRemoteDataManager methods	67
BSTR CurrentDirectory();	67
BOOL ChangeDirectory(LPCTSTR lpszNewDirectory);	67
BOOL FindFirst(LPCTSTR lpszPattern, BSTR FAR* pbstrFullQualName);	68
BOOL FindNext(BSTR FAR* pbstrFullQualName);	68
LPDISPATCH RetrieveObject(LPCTSTR lpszFQName, short wAccessMode, LPCTSTR lpszPassword);	69
LPDISPATCH RetrieveObjectEx(LPCTSTR lpszFQName, short wAccessMode, LPCTSTR lpszPassword);	69
LPDISPATCH CreateObject(LPCTSTR lpszFQName);	70
LPDISPATCH CreateObjectEx(LPCTSTR lpszFQName);	71
BOOL DeleteObject(LPCTSTR lpszFQName, LPCTSTR lpszPassword);	72
BOOL Launch(LPCTSTR lpszFQName);	72
CRemoteData object	73
Get and Set methods	73
Remote object Detail methods	75
Remote object methods	79
CRemoteDataEx object	82
Visual C++ sample code for remote functionality	82
CHostDataManager methods	83
BSTR CurrentDirectory();	83
BOOL ChangeDirectory(LPCTSTR lpszNewDirectory);	84
BOOL FindFirst(LPCTSTR lpszPattern, BSTR FAR* pbstrFullQualName);	84
BOOL FindNext(BSTR FAR* pbstrFullQualName);	85
LPDISPATCH RetrieveObject(LPCTSTR lpszFQName, short wAccessMode, LPCTSTR lpszPassword);	85
LPDISPATCH RetrieveObjectEx(LPCTSTR lpszFQName, short wAccessMode, LPCTSTR lpszPassword);	86
LPDISPATCH CreateObject(LPCTSTR lpszName);	87
LPDISPATCH CreateObjectEx(LPCTSTR lpszName);	87
BOOL DeleteObject(LPCTSTR lpszFQName, LPCTSTR lpszPassword);	88
BOOL Launch(LPCTSTR lpszFQName);	88
CHostData object	89
Get and Set methods	89
Host object Detail methods	90
Host object methods	95

CHostDataEx object99

 Visual C++ sample code for host functionality102

Awrem32 functions103

 boolean awConnect(BSTR FileName);103

 boolean awDisconnect();103

 boolean FileXferFromHost(BSTR HostFile, BSTR RemoteFile);104

 boolean FileXferToHost(BSTR HostFile, BSTR RemoteFile);104

 boolean CreateFolderOnHost(BSTR FolderName);105

 boolean ExecuteHostFile(BSTR FileName);105

 BSTR GetError();106

 short ConnectionStatus();106

Index

Using OLE Automation with Symantec pcAnywhere

This chapter includes the following topics:

- [About OLE Automation](#)
- [About the pcAnywhere Automation Server](#)
- [What you can do with the pcAnywhere Automation Server](#)
- [Before you start](#)
- [Where to find more information](#)

About OLE Automation

OLE Automation is a technology that lets you create an external application or other development tool (such as a script or macro) that can control and automate any exposed function within an application.

OLE Automation consists of the following components:

- OLE Automation server: An application or software component that exposes its functionality so that it can be accessed or controlled by other applications or development tools

The pcAnywhere Automation Server is an example of an OLE Automation server.

See [“About the pcAnywhere Automation Server”](#) on page 8.

- OLE Automation controller: An application or development tool that accesses and controls the components that have been exposed by the OLE Automation server

You can use any programming language that supports OLE Automation.

The two most common programming languages are Microsoft Visual Basic and Microsoft Visual C++.

An external application accesses an OLE Automation server by 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. After an application obtains an interface to the server, it can then call any internal interface method as though it were part of the external application.

About the pcAnywhere Automation Server

The pcAnywhere Automation Server lets external applications manage pcAnywhere remote, host, and caller information files to automate remote control and file transfer tasks. The pcAnywhere Automation Server functions as a programmable replacement for the Symantec pcAnywhere user interface and mirrors much of its default behavior.

For example, when you create a host object in pcAnywhere, the first available modem TAPI device is assigned by default. Similarly, when you create a host object using the pcAnywhere Automation Server and then enumerate through the list of assigned connections, the first available modem TAPI device is already assigned.

What you can do with the pcAnywhere Automation Server

The pcAnywhere Automation Server lets you automate a variety of administrative and productivity tasks. For example, you can do the following:

- Automatically distribute and install software updates on multiple computers across your network.
- Schedule automatic file transfers between computers for audit or archive purposes.
- Automatically add a name to or remove a name from the allowed callers list on every pcAnywhere host on your network.

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

For more information, see [“Visual Basic object definitions”](#) on page 15 and [“Visual C++ object definitions”](#) on page 65.

Before you start

During a connection to the pcAnywhere Automation Server and its interfaces, identifier parameters, known as globally unique identifiers (GUIDs), are passed to the automation library API functions. A separate GUID is assigned to the pcAnywhere Automation Server and to each exposed interface. These GUIDs must be present in the system registry to connect an external application to the pcAnywhere Automation Server and its interfaces.

If you are running the external application on a computer on which Symantec pcAnywhere is installed, you can register the GUID entries automatically.

Automatically registering the remote engine

Before connecting to another computer for the first time using your OLE client, you must self-register the remote engine. You can do this automatically by starting a remote object using Symantec pcAnywhere.

To automatically register the remote engine

- 1 To open Symantec pcAnywhere, do one of the following:
 - On the desktop, double-click the Symantec pcAnywhere program icon.
 - On the Windows taskbar, click **Start > Programs > Symantec pcAnywhere**.
- 2 In the pcAnywhere Manager window, click **Remotes**.
- 3 Double-click a remote icon.

This process registers the remote engine. You do not need to complete the connection.

Manually registering the remote engine

If pcAnywhere is not installed on the computer on which you are running the external application, you must register the GUIDs manually by running the pcAnywhere Automation Server executable file (Winawsvr.exe). You only need to run the executable file once to add the GUIDs to the registry. The Winawsvr.exe file is located in the default Symantec pcAnywhere data directory.

Accessing the pcAnywhere Automation Server

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

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.

Accessing the pcAnywhere Automation Server with Visual Basic

The Visual Basic programming language has built-in support for interacting with OLE Automation servers such as the pcAnywhere Automation Server. You create a Standard Exe project, then enter code in each method to access the pcAnywhere Automation Server. Visual Basic takes the high-level method calls in the source files and expands them internally into the corresponding low-level OLE Automation method calls.

See [“Visual Basic object definitions”](#) on page 15.

To access the pcAnywhere Automation Server with Visual Basic

- 1 Add a pair of Object variables for each pcAnywhere object that you want 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 set the current phone number, place the property name to the left of the assignment operator.
For example, `RemoteData.PhoneNumber = "555-1212"`

Accessing the pcAnywhere Automation Server with Visual C++

The pcAnywhere Automation Server uses type libraries to expose information about its interfaces and methods to automation clients that are written in Visual C++. These type libraries use Microsoft Foundation Classes (MFC), which can be imported into your application using the Visual C++ ClassWizard.

The data manager classes that are provided in the type libraries provide the functionality that is needed to obtain an interface to the pcAnywhere Automation Server and perform 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.

See [“Visual C++ object definitions”](#) on page 65.

Importing and viewing classes

The pcAnywhere Automation Server uses the following type libraries:

- Winawsvr.tlb: Provides the information needed to connect to the pcAnywhere Automation Server and access its interfaces
- Awrem32.tlb: Provides the information needed to control pcAnywhere connections

Import and view classes

The following procedures explain how to import the class definitions from the pcAnywhere Automation Server type libraries into your MFC application and then view the classes that have been added to your application.

To import classes

- 1 In Visual C++, create an MFC application.
- 2 On the View menu, click **Class Wizard**.
- 3 In the Class Wizard dialog box, click **Add Class**, then click **From a type library**.
- 4 Double-click **winawsvr.tlb**.
- 5 In the Confirm Classes dialog box, click **OK** to import all class definitions.
- 6 In the Class Wizard dialog box, click **Add Class**, then click **From a type library**.
- 7 Double-click **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 process.
The classes are added to the application. These classes allow you to manipulate objects and manage connections.
Importing the class definitions from the type libraries also adds support files to the application. These files contain the class definitions and implementation source code for the pcAnywhere Automation Server.
See [“Viewing the class definitions and implementation files”](#) on page 13.

To view the added classes

- 1 In Visual C++, open your MFC application.
- 2 In the Workspace window, click the **ClassView** tab.

Viewing the class definitions and implementation files

When you import the pcAnywhere Automation Server type libraries into your application, the following files are added:

- Winawsvr.h
- Winawsvr.cpp
- Awrem32.h
- Awrem32.cpp

These files contain the class definitions and implementation source code for the pcAnywhere Automation Server. You do not need to edit these files; however, each application source file that contains calls to the pcAnywhere Automation Server methods must include Winawsvr.h.

To view the class definitions and implementation files

- 1 In Visual C++, open your MFC application.
- 2 In the Workspace window, click the **FileView** tab.

Launching host and remote OLE objects

Symantec pcAnywhere requires that all host and remote objects be stored in the default data directory. Before launching a host or remote object that you created using OLE, ensure that it is located in the pcAnywhere default data directory.

Where to find more information

For more information, see the following references:

- Blaszcak, 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.

Visual Basic object definitions

This chapter includes the following topics:

- [About Visual Basic objects](#)
- [CRemoteDataManager methods](#)
- [CRemoteData properties](#)
- [CRemoteDataEx object](#)
- [CHostDataManager methods](#)
- [CHostData properties](#)
- [CHostDataEx object](#)
- [Awrem32 functions](#)

About Visual Basic objects

The pcAnywhere Automation Server provides the following components to support OLE Automation:

- Winawsvr: Provides the information needed to connect to the pcAnywhere Automation Server and access its interfaces
- Awrem32: Provides the information needed to control pcAnywhere connections

Table 2-1 describes the objects that comprise Winawsvr.

Table 2-1 Winawsvr objects

Object	Description	Reference
CRemoteDataManager	Provides the methods for creating, opening, modifying, saving, and deleting CRemoteData objects	See “CRemoteDataManager methods” on page 17.
CRemoteData	Defines the parameters for accessing and controlling pcAnywhere remote functionality	See “CRemoteData properties” on page 22 and “CRemoteDataEx object” on page 32.
CHostDataManager	Provides the methods for creating, opening, modifying, saving, and deleting CHostData objects	See “CHostDataManager methods” on page 35.
CHostData	Defines the parameters for accessing and controlling pcAnywhere host functionality	See “CHostData properties” on page 40 and “CHostDataEx object” on page 52.

Awrem32 has one object, which consists of eight interfaces to support remote control and file transfer sessions.

See “Awrem32 functions” on page 61.

Some functions, including the Gateway functionality, are no longer supported. However, object definitions are provided for use with earlier versions.

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

CRemoteDataManager methods

The CRemoteDataManager methods provide the parameters and return values for accessing and controlling CRemoteData objects.

CurrentDirectory()

Returns the full path name of the current directory in which pcAnywhere remote objects are stored. [Table 2-2](#) defines the CurrentDirectory() return value.

Table 2-2 CurrentDirectory() return value

Return value	Description
String	The full path name of the current pcAnywhere data directory

ChangeDirectory(NewDirectory)

Changes the current directory in which pcAnywhere remote objects are stored. [Table 2-3](#) defines the ChangeDirectory parameter.

Table 2-3 ChangeDirectory parameter

Parameter	Description
NewDirectory	Name of an existing directory

[Table 2-4](#) defines the ChangeDirectory return value.

Table 2-4 ChangeDirectory return value

Return value	Description
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. [Table 2-5](#) defines the FindFirst parameters.

Table 2-5 FindFirst parameters

Parameter	Description
Pattern as string	File name pattern to filter object files (an asterisk [*] finds all files in the current directory)
Name as string	Return buffer for the full path name of the remote object file that matches the specified pattern

[Table 2-6](#) defines the FindFirst return value.

Table 2-6 FindFirst return value

Return value	Description
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 that matches the pattern, if any. [Table 2-7](#) defines the FindNext parameter.

Table 2-7 FindNext parameter

Parameter	Description
Name as string	Return buffer for the full path name of the remote object file that matches the pattern specified in the original call to FindFirst()

[Table 2-8](#) defines the FindNext return value.

Table 2-8 FindNext return value

Return value	Description
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. [Table 2-9](#) defines the RetrieveObject parameters.

Table 2-9 RetrieveObject parameters

Parameter	Description
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.

[Table 2-10](#) defines the RetrieveObject return value.

Table 2-10 RetrieveObject return value

Return value	Description
Object	CRemoteData object from the specified file

RetrieveObjectEx(Name, AccessMode, Password)

Retrieves a CRemoteDataEx object by file name. [Table 2-11](#) defines the RetrieveObjectEx parameters.

Table 2-11 RetrieveObjectEx parameters

Parameter	Description
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.

[Table 2-12](#) defines the RetrieveObjectEx return value.

Table 2-12 RetrieveObjectEx return value

Return value	Description
Object	CRemoteDataEx object from the specified file

CreateObject(Name)

Creates a CRemoteData object and returns an LPDISPATCH pointer to it. [Table 2-13](#) defines the CreateObject parameter.

Table 2-13 CreateObject parameter

Parameter	Description
Name as string	The fully qualified remote object file name for the new object

[Table 2-14](#) defines the CreateObject return value.

Table 2-14 CreateObject return value

Return value	Description
Object	CRemoteData

CreateObjectEx(Name)

Creates a CRemoteDataEx object and returns an LPDISPATCH pointer to it. [Table 2-15](#) defines the CreateObjectEx parameter.

Table 2-15 CreateObjectEx parameter

Parameter	Description
Name as string	The fully qualified remote object file name for the new object

[Table 2-16](#) defines the CreateObjectEx return value.

Table 2-16 CreateObjectEx return value

Return value	Description
Object	CRemoteDataEx

DeleteObject(Name, Password)

Deletes a remote object file. [Table 2-17](#) defines the DeleteObject parameters.

Table 2-17 DeleteObject parameters

Parameter	Description
Name as string	The fully qualified remote object file name of the object to be deleted
Password as string	Object password

[Table 2-18](#) defines the DeleteObject return value.

Table 2-18 DeleteObject return value

Return value	Description
Boolean	TRUE if object is deleted

CRemoteData properties

Table 2-19 defines the properties and parameters that are available for the CRemoteData object. Replace the information in angle brackets with the actual values.

Table 2-19 CRemoteData properties and parameters

Property	Parameter	Description
<CRemoteData>.ComputerName(String)	String	Sets the computer name or IP address of the host computer.
<CRemoteData>.PhoneNumber(String)	String	Sets the phone number of the host computer.
<CRemoteData>.UseDialingProperties(Bool)	Bool	Sets the system dialing properties.
<CRemoteData>.RedialCount(Integer)	Integer	Sets the number of redial attempts before cancelling the call.
<CRemoteData>.RedialDelay(Integer)	Integer	Sets the number of seconds to wait between redial attempts.
<CRemoteData>.AutoLoginName(String)	String	Sets the name of the user for automatic login. For more information about using domain logins, see “CRemoteDataEx object” on page 32.
<CRemoteData>.AutoLoginPassword(String)	String	Sets the password for automatic logins in the remote object. For security reasons, the pcAnywhere Automation Server does not provide the ability to read the password value. A password value is not returned.
<CRemoteData>.Password(String)	String	Sets the password on the remote object for use with the ExecuteProtection, ReadProtection, and WriteProtection settings. For security reasons, the pcAnywhere Automation Server does not provide the ability to read the password value. A password value is not returned.
<CRemoteData>.ExecuteProtection(Bool)	Bool	Sets the requirement of a password to execute the object. Set by Password.
<CRemoteData>.ReadProtection(Bool)	Bool	Sets the requirement of a password to view the properties of the remote object. Set by Password.
<CRemoteData>.WriteProtection(Bool)	Bool	Sets the requirement of a password to save changes to the remote object. Set by Password.

Table 2-19 CRemoteData properties and parameters

Property	Parameter	Description
<CRemoteData>.LogSession(Bool)	Bool	Activates and deactivates session logging.
<CRemoteData>.RecordFile(String)	String	Sets the fully qualified path and name to the location of the file that records the active session.
<CRemoteData>.RecordSession(Bool)	Bool	Activates and deactivates automatic session recording.
<CRemoteData>.ReadObject(String)	String	Sets the password of the object. This can be used to refresh the local data copy of the remote object.
<CRemoteData>.WriteObject(String)	String	Sets the password of the object. This is used to create the remote object or to write changes that you have made to the remote object.

[Table 2-20](#) defines the properties and return values for CRemoteData. Replace the information in angle brackets with the actual values.

Table 2-20 CRemoteData properties and return values

Property	Return value	Description
String = <CRemoteData>.ComputerName	String	Returns the computer name or IP address of the host computer
String = <CRemoteData>.PhoneNumber	String	Returns the phone number of the host computer
Bool = <CRemoteData>.UseDialingProperties	Bool	Returns the system dialing properties that are set in the remote object
Integer = <CRemoteData>.RedialCount	Integer	Returns the number of redial attempts that is set in the remote object
Integer = <CRemoteData>.RedialDelay	Integer	Returns the number of seconds between redial attempts
String = <CRemoteData>.AutoLoginName	String	Returns the login name that is used for automatic logins
Bool = <CRemoteData>.ExecuteProtection	Bool	Returns the value of the ExecuteProtection setting
Bool = <CRemoteData>.ReadProtection	Bool	Returns the value of the ReadProtection setting
Bool = <CRemoteData>.WriteProtection	Bool	Returns the value of the WriteProtection setting
Bool = <CRemoteData>.LogSession	Bool	Returns TRUE if session logging is enabled

Table 2-20 CRemoteData properties and return values

Property	Return value	Description
String = <CRemoteData>.RecordFile	String	Returns the fully qualified path and name of the session recording file
Bool = <CRemoteData>.RecordSession	Bool	Returns the value of the session recording setting

Connection type properties

Table 2-21 defines the connection type properties and parameters. Replace the information in angle brackets with the actual values.

Table 2-21 Connection type properties and parameters

Property	Parameter	Description
<CRemoteData>.ConnectionType(String)	String	Sets the connection type of the remote. The value that is passed in must be a valid connection type as defined by the FirstConnectionType() and NextConnectionType() functions. The following are examples of valid connection types: ■ COM1 ■ COM2 ■ COM3 ■ COM4 ■ LPT1 ■ LPT2 ■ LPT3 ■ LPT4 ■ TCP/IP ■ SPX ■ NetBIOS ■ Infrared ■ ISDN via CAPI 2.0 ■ Modem name (as it appears on the computer)

Table 2-22 defines the connection type properties and return values. Replace the information in angle brackets with the actual values.

Table 2-22 Connection type properties and return values

Property	Return value	Description
String = <CRemoteData>.ConnectionTypes	String	Returns the connection type of the remote object.
Integer = <CRemoteData>.ConnectionType	Integer	Returns the number of available connection types. The following are examples of valid connection types: ■ COM1 ■ COM2 ■ COM3 ■ COM4 ■ LPT1 ■ LPT2 ■ LPT3 ■ LPT4 ■ TCP/IP ■ SPX ■ NetBIOS ■ Infrared ■ ISDN via CAPI 2.0 ■ Modem name (as it appears on the computer)
String = <CRemoteData>.FirstConnectionType	String	Returns the first available connection type.
String = <CRemoteData>.NextConnectionType	String	Returns the next available connection type. This is called sequentially for the number of connection types that is set in <CRemoteData>.ConnectionType to enumerate all connection types.
Bool = <CRemoteData>.FindConnectionType(ConnectionType)	Bool	Returns TRUE if the named connection type is found in the list of available connection types.

Dialing properties

Table 2-23 defines the properties and parameters for setting the dialing properties for modem connections. Replace the information in angle brackets with the actual values.

Table 2-23 Properties and parameters for dialing properties

Property	Parameter	Description
<CRemoteData>.AreaCode(String)	String	Sets the area code dialing properties for modem connections
<CRemoteData>.CountryCode(String)	String	Sets the country code dialing properties for modem connections

Table 2-24 defines the properties and return values for modem dialing properties. Replace the information in angle brackets with the actual values.

Table 2-24 Properties and return values for dialing properties

Property	Return value	Description
String = <CRemoteData>.AreaCode	String	Returns the area code dialing properties.
String = <CRemoteData>.CountryCode	String	Returns the dialing properties country code.
Integer = <CRemoteData>.CountryCodes	Integer	Returns the number of available country codes.
String = <CRemoteData>.FirstCountryCode	String	Returns the first available country code that is listed in the operating system.
String = <CRemoteData>.NextCountryCode	String	Returns the next available country code. This is called sequentially for the number of country codes that is set in <CRemoteData.CountryCodes> to enumerate all country codes.

COM device properties

[Table 2-25](#) describes the properties and parameters that let you customize the port settings for modem and other COM-based connections. Replace the information in angle brackets with the actual values.

Table 2-25 COM device properties and parameters

Property	Parameter	Description
<CRemoteData>.ComParity(String)	String	Sets the communications parity The following values are valid: ■ <None> ■ Odd ■ Even ■ Mark ■ Space
<CRemoteData>.ComFlowControl(String)	String	Sets the flow control of COM-based connection types The following values are valid: ■ <None> ■ XONXOFF ■ RTS/CTS ■ Both
<CRemoteData>.ComStartedBy(String)	String	Sets the start setting of COM-based connection types The following values are valid: ■ Always connected ■ Carrier detect (DCD) ■ Clear to send (CTS) ■ Data set ready (DSR) ■ Ring indicator (RI) ■ Receive 2 <CR>'s ■ Modem response

Table 2-25COM device properties and parameters

Property	Parameter	Description
<CRemoteData>.ComEndedBy(String)	String	Sets the end setting of COM-based connection types The following values are valid: ■ Always connected ■ Carrier detect (DCD) ■ Clear to send (CTS) ■ Data set ready (DSR) ■ Ring indicator (RI)
<CRemoteData>.ComSpeed(Long)	Long	Contains the maximum COM speed setting The following values are valid: ■ 110 ■ 300 ■ 600 ■ 1200 ■ 2400 ■ 4800 ■ 9600 ■ 38400 ■ 57600 ■ 115200

Table 2-26 describes the COM device properties and return values. Replace the information in angle brackets with the actual values.

Table 2-26 COM device properties and return values

Property	Return value	Description
String = <CRemoteData>.ComParity	String	Returns one of the following values for communications parity: ■ <None> ■ Odd ■ Even ■ Mark ■ Space
String = <CRemoteData>.ComFlowControl	String	Returns the Com Flow setting of the remote object The following values are valid: ■ <None> ■ XONXOFF ■ RTS/CTS ■ Both
String = <CRemoteData>.ComStartedBy	String	Returns the Com Start control of COM-based connection types The following values are valid: ■ Always connected ■ Carrier detect (DCD) ■ Clear to send (CTS) ■ Data set ready (DSR) ■ Ring indicator (RI) ■ Receive 2 <CR>'s ■ Modem response
String = <CRemoteData>.ComEndedBy	String	Returns the Com End control of COM-based connection types The following values are valid: ■ Always connected ■ Carrier detect (DCD) ■ Clear to send (CTS) ■ Data set ready (DSR) ■ Ring indicator (RI)

Table 2-26COM device properties and return values

Property	Return value	Description
Long = <CRemoteData>.ComSpeed	Long	Returns the current setting of the Com Speed of the remote object The following values are valid: ■ 110 ■ 300 ■ 600 ■ 1200 ■ 2400 ■ 4800 ■ 9600 ■ 38400 ■ 57600 ■ 115200

NetBIOS device properties

Table 2-27 defines the property and parameter for a NetBIOS network device. Replace the information in angle brackets with the actual values.

Table 2-27NetBIOS property and parameter

Property	Parameter	Description
<CRemoteData>.LanaNumber(Integer)	Integer	Sets the LAN Adapter (LANA) number for NetBIOS connections

Table 2-28 defines the NetBIOS property and return value. Replace the information in angle brackets with the actual values.

Table 2-28NetBIOS return value

Property	Return value	Description
Integer = <CRemoteData>.LanaNumber	Integer	Returns the current setting of the LAN Adapter (LANA) number for NetBIOS connections

ISDN via CAPI 2.0 device properties

[Table 2-29](#) defines the properties and parameters for European ISDN connections. Replace the information in angle brackets with the actual values.

Table 2-29 ISDN via CAPI 2.0 properties and parameters

Property	Parameter	Description
<CRemoteData>.CapiChannelBonding(Bool)	Bool	Activates or deactivates channel bonding for ISDN CAPI devices
<CRemoteData>.CapiExtensions(String)	String	Sets any additional CAPI extensions that are needed for communications

[Table 2-30](#) defines the properties and return values for European ISDN connections. Replace the information in angle brackets with the actual values.

Table 2-30 ISDN via CAPI 2.0 properties and return values

Property	Return value	Description
Bool= <CRemoteData>.CapiChannelBonding	Bool	Returns the current ISDN CAPI channel bonding setting in the remote object
String = <CRemoteData>.CapiExtensions	String	Returns the current list of CAPI extensions from the remote object

Network (TCP/IP, SPX) device properties for Gateways

The following properties can be used only with pcAnywhere 9.2x:

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

Note: Later versions of pcAnywhere do not support the Gateway functionality.

CRemoteDataEx object

The CRemoteDataEx object contains the same functionality as the CRemoteData object with some additional functionality. [Table 2-31](#) describes the properties and parameters. Replace the information in angle brackets with the actual values.

Table 2-31 CRemoteDataEx parameters

Property	Parameter	Description
<CRemoteData>.PrivateKey(String)	String	Sets the name of the private key container to use.
<CRemoteData>.CertificateName(String)	String	Sets the common name of the private key to use.
<CRemoteData>.EncryptionLevel(Byte)	Byte	Sets the encryption level. The following values are valid: ■ -1: None ■ 0: pcAnywhere ■ 1: Symmetric ■ 2: Public key
<CRemoteData>.DenyLowerEncrypt(Bool)	Bool	Defines whether the remote computer allows a connection to a host computer that uses a lower level of encryption.
<CRemoteData>.AutoDomain(String)	String	Sets the domain name for automatic logins. This option is used with NT authentication and Windows authentication.

[Table 2-32](#) describes the properties and return values for the CRemoteDataEx object. Replace the information in angle brackets with the actual values.

Table 2-32 CRemoteDataEx properties and return values

Property	Return value	Description
String = <CRemoteData>.PrivateKey	String	Returns the name of the currently active private key container.
String = <CRemoteData>.CertificateName	String	Returns the common name of the active private key container.

Table 2-32 CRemoteDataEx properties and return values

Property	Return value	Description
Byte = <CRemoteData>.EncryptionLevel	Byte	Returns one of the following encryption settings: ■ -1: None ■ 0: pcAnywhere ■ 1: Symmetric ■ 2: Public key
Bool = <CRemoteData>.DenyLowerEncrypt	Bool	Returns the value of the deny lower encryption setting.
String = <CRemoteData>.AutoDomain	String	Returns the domain name setting for automatic logins. This option is used with NT authentication and Windows authentication.

Visual Basic sample code for remote functionality

The following 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 the remote file in a directory as follows:

```

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 the connection type to TCP/IP and the computer name "Host1," and then launch it as follows:

```
Private Sub Command6_Click()  
    Dim RemoteDataManager as Object  
    Dim RemoteData as Object  
    Dim s as string  
  
    Set RemoteDataManager =  
        CreateObject("WINAHSV.RMOTEDATAMANAGER")  
    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

The CHostDataManager methods provide the parameters and return values for accessing and controlling CHostData objects.

CurrentDirectory()

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

Table 2-33 CurrentDirectory() return value

Return value	Description
String	The full path name of the current pcAnywhere data directory

FindFirst(Pattern, Name string)

Finds the first pcAnywhere host object file (*.bhf) in the current directory, based on the specified file name pattern. [Table 2-34](#) defines the parameters for FindFirst.

Table 2-34 FindFirst parameters

Parameter	Description
Pattern as string	File name pattern to filter object files (an asterisk [*] finds all files in the current directory)
Name as string	Return buffer for the full path name of the host object file that matches the specified pattern

[Table 2-35](#) defines the return value for FindFirst.

Table 2-35 FindFirst return value

Return value	Description
Boolean	TRUE if a host 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 host object file in the current directory, FindNext() can be called to find the next file that matches the pattern, if any. [Table 2-36](#) defines the parameter for FindNext.

Table 2-36 FindNext parameter

Parameter	Description
Name as string	Return buffer for the full path name of the host object file that matches the pattern specified in the original call to FindFirst()

[Table 2-37](#) defines the return value for FindNext.

Table 2-37 FindNext return value

Return value	Description
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. [Table 2-38](#) defines the parameters for RetrieveObject.

Table 2-38 RetrieveObject parameters

Parameter	Description
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.

[Table 2-39](#) defines the return value for RetrieveObject.

Table 2-39 RetrieveObject return value

Return value	Description
Object	CHostData object from the specified file

RetrieveObjectEx(Name, AccessMode, Password)

Retrieves a CHostDataEx object by file name. [Table 2-40](#) defines the parameters for RetrieveObjectEx.

Table 2-40 RetrieveObjectEx parameters

Parameter	Description
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.

[Table 2-41](#) defines the return value for RetrieveObjectEx.

Table 2-41 RetrieveObjectEx return value

Return value	Description
Object	CHostDataEx object from the specified file

CreateObject(Name)

Creates a CHostData object and returns an LPDISPATCH pointer to it. [Table 2-42](#) defines the parameter for CreateObject.

Table 2-42 CreateObject parameter

Parameter	Description
Name as string	The fully qualified host object file name for the new object

[Table 2-43](#) defines the return value for CreateObject.

Table 2-43 CreateObject return values

Return value	Description
Object	CHostData

CreateObjectEx(Name)

Creates a CHostDataEx object and returns an LPDISPATCH pointer to it. [Table 2-44](#) defines the parameter for CreateObjectEx.

Table 2-44 CreateObjectEx parameter

Parameter	Description
Name as string	The fully qualified host object file name for the new object

[Table 2-45](#) defines the return value for CreateObjectEx.

Table 2-45 CreateObjectEx return value

Return value	Description
Object	CHostDataEx

DeleteObject(Name, Password)

Deletes a host object file. [Table 2-46](#) defines the parameters for DeleteObject.

Table 2-46 DeleteObject parameters

Parameter	Description
Name as string	The fully qualified host object file name of the object to be deleted
Password as string	Object password

[Table 2-47](#) defines the return value for DeleteObject.

Table 2-47 DeleteObject return value

Return value	Description
Boolean	TRUE if object is deleted

Launch(Name)

Launches a host object file, which opens the pcAnywhere host terminal window.

Table 2-48 Launch parameter

Parameter	Description
Name as string	The fully qualified host object file of the object to be launched

Table 2-49 defines the Launch return value.

Table 2-49 Launch return value

Return value	Description
Boolean	TRUE if object is successfully launched

CHostData properties

Table 2-50 describes the properties and parameters that are available for the CHostData object. Replace the information in angle brackets with the actual values.

Table 2-50 CHostData properties and parameters

Property	Parameter	Description
<CHostData>.PhoneNumber(String)	String	Sets the phone number of the host computer.
<CHostData>.UseDialingProperties(Bool)	Bool	Sets the system dialing properties.
<CHostData>.RedialCount(Integer)	Integer	Sets the number of redial attempts before cancelling the call.
<CHostData>.RedialDelay(Integer)	Integer	Sets the number of seconds to wait between redial attempts.
<CHostData>.Password(String)	String	Sets the password on the host object for use with the ExecuteProtection, ReadProtection, and WriteProtection settings. For security reasons, the pcAnywhere Automation Server does not provide the ability to read the password value. A password value is not returned.
<CHostData>.ExecuteProtection(Bool)	Bool	Sets the requirement of a password to execute the object. Set by Password.

Table 2-50 CHostData properties and parameters

Property	Parameter	Description
<CHostData>.ReadProtection(Bool)	Bool	Sets the requirement of a password to view the properties of the host object. Set by Password.
<CHostData>.WriteProtection(Bool)	Bool	Sets the requirement of a password to save changes to the host object. Set by Password.
<CHostData>.LogSession(Bool)	Bool	Activates and deactivates session logging.
<CHostData>.RecordFile(String)	String	Sets the fully qualified path and name to the location of the file that records the active session.
<CHostData>.RecordSession(Bool)	Bool	Activates and deactivates automatic session recording.
<CHostData>.ReadObject(String)	String	Sets the password for the object. This can be used to refresh the local data copy of the host object.
<CHostData>.WriteObject(String)	String	Sets the password for the object. This is used to create the host object or to write changes that you have made to the host object.

[Table 2-51](#) describes the CHostData properties and return values. Replace the information in angle brackets with the actual values.

Table 2-51 CHostData properties and return values

Property	Return value	Description
String = <CHostData>.PhoneNumber	String	Returns the phone number of the host computer
Bool = <CHostData>.UseDialingProperties	Bool	Returns the use of the system dialing properties that are set in the host object
Integer = <CHostData>.RedialCount	Integer	Returns the number of redial attempts that is set in the host object
Integer = <CHostData>.RedialDelay	Integer	Returns the number of seconds between redial attempts
Bool = <CHostData>.ExecuteProtection	Bool	Returns the value of the ExecuteProtection setting
Bool = <CHostData>.ReadProtection	Bool	Returns the value of the ReadProtection setting
Bool = <CHostData>.WriteProtection	Bool	Returns the value of the Write Protection setting
Bool = <CHostData>.LogSession	Bool	Returns TRUE if session logging is enabled

Table 2-51 CHostData properties and return values

Property	Return value	Description
String = <CHostData>.RecordFile	String	Returns the fully qualified path and name of the session recording file
Bool = <CHostData>.RecordSession	Bool	Returns the value of the session recording setting

Connection type properties

Table 2-52 describes the connection type properties and parameters. Replace the information in angle brackets with the actual values.

Table 2-52 Connection type properties and parameters

Property	Parameter	Description
<CHostData>.ConnectionType(String)	String	Sets the connection type of the host. The value that is passed in must be a valid connection type as defined by the FirstConnectionType() and NextConnectionType() functions. The following are examples of valid connection types: ■ COM1 ■ COM2 ■ COM3 ■ COM4 ■ LPT1 ■ LPT2 ■ LPT3 ■ LPT4 ■ TCP/IP ■ SPX ■ NetBIOS ■ Infrared ■ ISDN via CAPI 2.0 ■ Modem name (as it appears on the computer)
<CHostData>.AssignConnection(String)	String	Sets the connection type that is defined in the string to active status.

Table 2-52 Connection type properties and parameters

Property	Parameter	Description
<CHostData>.UnassignConnection(String)	String	Sets the connection type that is defined in the string to inactive status.

[Table 2-53](#) describes the connection type properties and return values. Replace the information in angle brackets with the actual values.

Table 2-53 Connection type properties and return values

Property	Return value	Description
String = <CHostData>.ConnectionTypes	String	Returns the connection type of the host object.
Integer = <CHostData>.ConnectionType	Integer	Returns the number of available connection types. The following are examples of valid connection types: ■ COM1 ■ COM2 ■ COM3 ■ COM4 ■ LPT1 ■ LPT2 ■ LPT3 ■ LPT4 ■ TCP/IP ■ SPX ■ NetBIOS ■ Infrared ■ ISDN via CAPI 2.0 ■ Modem name (as it appears on the computer)
String = <CHostData>.FirstConnectionType	String	Returns the first available connection type.
String = <CHostData>.NextConnectionType	String	Returns the next available connection type. This is called sequentially for the number of connection types that is set in <CHostData>.ConnectionTypes to enumerate all connection types.

Table 2-53 Connection type properties and return values

Property	Return value	Description
Bool = <CHostData>.FindConnectionType (ConnectionType)	Bool	Returns TRUE if the named connection type is found in the list of available connection types.
Integer = <CHostData>.MaxAssignedConnections	Integer	Returns the maximum number of connection types that can be active on this host.
String = <CHostData>.FirstAssignedConnection	String	Returns the first assigned active connection type.
String = <CHostData>.NextAssignedConnection	String	Returns the next assigned active connection type.
Bool = <CHostData>.FindAssignedConnection (ConnectionType)	Bool	Returns TRUE if the connection type that is passed in matches any of the active connection types.

AssignConnection(ConnectionType) method

The AssignConnection(Connection Type) method 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 on 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 connection type that is passed in 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 support up to two assigned connection types. The AssignConnection method returns FALSE if it detects an attempt to exceed this limit.

[Table 2-54](#) defines the AssignConnection parameter.

Table 2-54 AssignConnection parameter

Parameter	Description
ConnectionType as string	The name of a connection device type to be assigned

[Table 2-55](#) defines the AssignConnection return value.

Table 2-55 AssignConnection return value

Return value	Description
Boolean	TRUE if this device type is available and the maximum allowed assigned connection count has not already been reached

UnassignConnection(ConnectionType) method

The UnassignConnection(Connection Type) method returns TRUE if the connection type that is passed in is successfully removed from the list of assigned connection types.

[Table 2-56](#) defines the UnassignConnection parameter.

Table 2-56 UnassignConnection parameter

Parameter	Description
ConnectionType as string	The name of a connection device type to be unassigned

[Table 2-57](#) defines the UnassignConnection return value.

Table 2-57 UnassignConnection return value

Return value	Description
Boolean	TRUE if this device type is successfully unassigned

Dialing properties

Table 2-58 defines the properties and return values for setting the dialing properties for modem connections. Replace the information in angle brackets with the actual values.

Table 2-58 Properties and parameters for modem dialing properties

Property	Parameter	Description
<CHostData>.AreaCode(String)	String	Sets the area code for the modem connections
<CHostData>.CountryCode(String)	String	Sets the country code for modem connections

Table 2-59 defines the properties and return values for modem dialing properties. Replace the information in angle brackets with the actual values.

Table 2-59 Properties and return values for modem dialing properties

Property	Return value	Description
String = <CHostData>.AreaCode	String	Returns the area code.
String = <CHostData>.CountryCode	String	Returns the country code.
Integer = <CHostData>.CountryCodes	Integer	Returns the number of available country codes.
String= <CHostData>.FirstCountryCode	String	Returns the first available country code that is listed in the operating system.
String = <CHostData>.NextCountryCode	String	Returns the next available country code. This is called sequentially for the number of country codes that is set in <CHostData>.CountryCodes to enumerate all country codes.

COM device properties

[Table 2-60](#) defines the properties and parameters that let you customize the port settings for modem and other COM-based connections. Replace the information in angle brackets with the actual values.

Table 2-60 COM device properties and parameters

Property	Parameter	Description
<CHostData>.ComParity(String)	String	Sets the communications parity The following values are valid: ■ <None> ■ Odd ■ Even ■ Mark ■ Space
<CHostData>.ComFlowControl(String)	String	Sets the flow control of COM-based connection types The following values are valid: ■ <None> ■ XONXOFF ■ RTS/CTS ■ Both
<CHostData>.ComStartedBy(String)	String	Sets the start setting of COM-based connection types The following values are valid: ■ Always connected ■ Carrier detect (DCD) ■ Clear to send (CTS) ■ Data set ready (DSR) ■ Ring indicator (RI) ■ Receive 2 <CR>'s ■ Modem response

Table 2-60COM device properties and parameters

Property	Parameter	Description
<CHostData>.ComEndedBy(String)	String	Sets the end setting of COM-based connection types The following values are valid: ■ Always connected ■ Carrier detect (DCD) ■ Clear to send (CTS) ■ Data set ready (DSR) ■ Ring indicator (RI)
<CHostData>.ComSpeed(Long)	Long	Sets the maximum COM speed setting The following values are valid: ■ 110 ■ 300 ■ 600 ■ 1200 ■ 2400 ■ 4800 ■ 9600 ■ 38400 ■ 57600 ■ 115200

Table 2-61 describes the COM device properties and return values. Replace the information in angle brackets with the actual values.

Table 2-61 COM device properties and return values

Property	Return value	Description
String = <CHostData>.ComParity	String	Returns one of the following values as the communications parity: ■ <None> ■ Odd ■ Even ■ Mark ■ Space
String = <CHostData>.ComFlowControl	String	Returns the Com Flow setting of the host object The following values are valid: ■ <None> ■ XONXOFF ■ RTS/CTS ■ Both
String = <CHostData>.ComStartedBy	String	Returns the Com Start control of COM-based connection types The following values are valid: ■ Always connected ■ Carrier detect (DCD) ■ Clear to send (CTS) ■ Data set ready (DSR) ■ Ring indicator (RI) ■ Receive 2 <CR>'s ■ Modem response
String = <CHostData>.ComEndedBy	String	Returns the Com End control of COM-based connection types The following values are valid: ■ Always connected ■ Carrier detect (DCD) ■ Clear to send (CTS) ■ Data set ready (DSR) ■ Ring indicator (RI)

Table 2-61 COM device properties and return values

Property	Return value	Description
Long = <CHostData>.ComSpeed	Long	Returns the current setting of the Com Speed of the host object The following values are valid: ■ 110 ■ 300 ■ 600 ■ 1200 ■ 2400 ■ 4800 ■ 9600 ■ 38400 ■ 57600 ■ 115200

NetBIOS device properties

[Table 2-62](#) defines the property and parameter for NetBIOS network devices. Replace the information in angle brackets with the actual values.

Table 2-62 NetBIOS property and parameter

Property	Parameter	Description
<CHostData>.LanaNumber(Integer)	Integer	Sets the LAN Adapter (LANA) number for NetBIOS connections

[Table 2-63](#) defines the property and return value for NetBIOS network devices. Replace the information in angle brackets with the actual values.

Table 2-63 NetBIOS return values

Property	Return value	Description
Integer = <CHostData>.LanaNumber	Integer	Returns the current setting of the LAN Adapter (LANA) number for NetBIOS connections

ISDN via CAPI 2.0 device properties

[Table 2-64](#) defines the properties and parameters for European ISDN connections. Replace the information in angle brackets with the actual values.

Table 2-64 ISDN via CAPI 2.0 properties and parameters

Property	Parameter	Description
<CHostData>.CapiChannelBonding(Bool)	Bool	Activates or deactivates channel bonding for ISDN CAPI devices
<CHostData>.CapiExtensions(String)	String	Sets any additional CAPI extensions that are needed for communications

[Table 2-65](#) defines the properties and return values for European ISDN connections.

Table 2-65 ISDN via CAPI 2.0 properties and return values

Property	Return value	Description
Bool= <CHostData>.CapiChannelBonding	Bool	Returns the current ISDN CAPI channel bonding setting in the host object
String = <CHostData>.CapiExtensions	String	Returns the current list of CAPI extensions from the host object

Network (TCP/IP, SPX) device properties for Gateways

The following properties can be used only with pcAnywhere 9.2x:

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

Note: Later versions of pcAnywhere do not support the Gateway functionality.

CHostDataEx object

The CHostDataEx object contains the same functionality as the CHostData object with some additional functionality. [Table 2-66](#) describes the properties and parameters. Replace the information in angle brackets with the actual values.

Table 2-66 CHostDataEx properties and parameters

Property	Parameter	Description
<CHostData>.CryptPrivateKey(String)	String	Sets the name of the private key container to use.
<CHostData>.CryptCommonName(String)	String	Sets the common name of the private key to use.
<CHostData>.CryptReqLevel(Byte)	Byte	Sets the encryption level. The following values are valid: ■ -1: None ■ 0: pcAnywhere ■ 1: Symmetric ■ 2: Public key
<CHostData>.CryptRefuseLower(Bool)	Bool	Defines whether the host computer accepts connections from a remote computer that uses a lower level of encryption.
<CHostData>.CallersPath(String)	String	Sets the fully qualified path to the caller files.
<CHostData>.ConfirmConnect(Bool)	Bool	Defines whether the host user will be prompted to confirm connections.
<CHostData>.ConfirmTimeout(Byte)	Byte	Sets the number of seconds before the prompt to confirm the connection time limit expires.
<CHostData>.ConfirmDeny(Bool)	Bool	Defines whether the connection should be ended if the prompt to confirm the connection time limit expires.
<CHostData>.PwCaseSensitive(Bool)	Bool	Forces the use of case-sensitive passwords.
<CHostData>.PwAttempts(Byte)	Byte	Sets the number of consecutive failed logon attempts that are allowed before ending the connection.
<CHostData>.PwTimeout(Byte)	Byte	Sets the number of minutes that a user has to complete the logon before the connection is dropped.

Table 2-66 CHostDataEx properties and parameters

Property	Parameter	Description
<CHostData>.ActiveKbds(Byte)	Byte	Defines which mouse and keyboard will be active during the connection. The following values are valid: ■ 0: Host and remote ■ 1: Host ■ 2: Remote
<CHostData>.InactiveTimeout(Byte)	Byte	Sets the number of minutes in which the keyboard and mouse can be inactive before the connection is ended.
<CHostData>.LockSystemWhileWait(Bool)	Bool	Sets the lock host computer upon startup setting.
<CHostData>.MinimizeOnLaunch(Bool)	Bool	Sets the run minimized host startup option.
<CHostData>.RunAsService(Bool)	Bool	Enables the host to run as a service.
<CHostData>.ConnLostWait(Byte)	Byte	Sets the number of minutes to wait before allowing another caller to connect.
<CHostData>.ConnLostHostOpts(Bool)	Bool	Defines whether to wait for another connection or cancel the host if the session ends abnormally. The following values are valid: ■ FALSE: Wait ■ TRUE: Cancel host
<CHostData>.EnableConnLostSecurity(Bool)	Bool	Activates or deactivates the end of session security options for sessions that end abnormally.

Table 2-66 CHostDataEx properties and parameters

Property	Parameter	Description
<CHostData>.AuthenticationType(Byte)	Byte	Sets the authentication type. The following values are valid: ■ 0: pcAnywhere ■ 1: pcAnywhere ■ 2: Windows ■ 3: NT ■ 4: pcAnywhere ■ 5: pcAnywhere ■ 6: ADS Active Directory Services ■ 7: Microsoft LDAP ■ 8: FTP ■ 9: HTTP ■ 10: HTTPS ■ 11: Netscape LDAP ■ 12: Novell LDAP ■ 13: RSA SecurID pcAnywhere authentication is used by default if no authentication value is set or if the set value is not valid (for example, the authentication type is not available).
<CHostData>.ConnLostSecurity(Byte)	Byte	Sets the security options for handling an abnormal end of session. The following values are valid: ■ 1: Log off user ■ 2: Restart host computer ■ 3: Lock computer
<CHostData>.CallbkDelay(Byte)	Byte	Sets the number of seconds to wait before the host modem calls back the remote.
<CHostData>.EndSessHostOpts(Bool)	Bool	Defines whether the host waits for another connection or is cancelled after a normal end of session. The following values are valid: ■ FALSE: Wait for next connection ■ TRUE: Cancel host

Table 2-66 CHostDataEx properties and parameters

Property	Parameter	Description
<CHostData>.EnableEndSessSecurity(Bool)	Bool	Activates or deactivates the security options for a normal end of session.
<CHostData>.EndSessSecurity(Byte)	Byte	Sets the security options for a normal end of session. The following values are valid: ■ 1: Log off user ■ 2: Restart host computer ■ 3: Lock computer
<CHostData>.BlankHost(Bool)	Bool	Sets screen blanking on the host computer. Some video cards do not support this option.
<CHostData>.AllowRemoteMouse(Bool)	Bool	Disables the use of the mouse on the remote computer during a session.
<CHostData>.RebootOnDisconnect(Bool)	Bool	Forces the computer to restart after any end of session if TRUE.
<CHostData>.PasswordAfterDisc(Bool)	Bool	Logs off the user after the session ends.
<CHostData>.LogFailures(Bool)	Bool	Logs failed password attempts.
<CHostData>.AllowDriveSecurity(Bool)	Bool	Enables drive security options. This setting is valid only on NTFS file systems.
<CHostData>.UseDirectoryServices(Bool)	Bool	Enables the use of directory services for authentication.
<CHostData>.DirectoryServiceEntry(String)	String	Sets the directory services settings.

Table 2-67 describes the properties and return values for the CHostDataEx object. Replace the information in angle brackets with the actual values.

Table 2-67 CHostDataEx properties and return values

Property	Return value	Description
String = <CHostDataEx>.CryptPrivateKey	String	Returns the currently active private key container.
String = <CHostDataEx>.CryptCommonName	String	Returns the common name of the active private key container.
Byte = <CHostDataEx>.CryptReqLevel	Byte	Returns one of the following encryption settings: ■ -1: None ■ 0: pcAnywhere ■ 1: Symmetric ■ 2: Public key
Bool = <CHostDataEx>.CryptRefuseLower	Bool	Returns the value of the deny lower encryption setting.
String = <CHostDataEx>.CallersPath	String	Returns the currently active path to the caller files.
Bool = <CHostDataEx>.ConfirmConnect	Bool	Returns the value of the confirm connection setting.
Byte = <CHostDataEx>.ConfirmTimeout	Byte	Returns the value of the deny lower encryption setting.
Bool = <CHostDataEx>.ConfirmDeny	Bool	Returns the value of the disconnect if timeout setting.
Bool = <CHostDataEx>.PwCaseSensitive	Bool	Returns the value of the make passwords case sensitive setting.
Byte = <CHostDataEx>.PwAttempts	Byte	Returns the value of the limit login attempts per call setting.
Byte = <CHostDataEx>.PwTimeout	Byte	Returns the value of the limit time to complete login setting (in minutes).

Table 2-67 CHostDataEx properties and return values

Property	Return value	Description
Byte = <CHostDataEx>.ActiveKbds	Byte	Returns the active keyboard and mouse settings. The following values are valid: ■ 0: Host and remote ■ 1: Host ■ 2: Remote
Byte = <CHostDataEx>.InactiveTimeout	Byte	Returns the number of minutes to wait before disconnecting if the inactivity time limit expires.
Bool = <CHostDataEx>.LockSystemWhileWait	Bool	Returns the lock host computer upon startup setting.
Bool = <CHostDataEx>.MinimizeOnLaunch	Bool	Returns the run minimized on host startup setting.
Bool = <CHostDataEx>.RunAsService	Bool	Returns the run host as a service setting.
Byte = <CHostDataEx>.ConnLostWait	Byte	Returns the number of minutes to wait before allowing another connection.
Bool = <CHostDataEx>.ConnLostHostOpts	Bool	Returns whether to wait for another connection or cancel the host if the session ends abnormally. The following values are valid: ■ FALSE: Wait ■ TRUE: Cancel host
Bool = <CHostDataEx>.ConnLostWait	Bool	Returns the value of the security option that is set for handling an abnormal end of session.

Table 2-67 CHostDataEx properties and return values

Property	Return value	Description
Byte = <CHostDataEx>.AuthenticationType	Byte	Returns the number reference of the authentication type. The following values are valid: ■ 0: pcAnywhere ■ 1: pcAnywhere ■ 2: Windows ■ 3: NT ■ 4: pcAnywhere ■ 5: pcAnywhere ■ 6: ADS Active Directory Services ■ 7: Microsoft LDAP ■ 8: FTP ■ 9: HTTP ■ 10: HTTPS ■ 11: Netscape LDAP ■ 12: Novell LDAP ■ 13: RSA SecurID pcAnywhere authentication is used by default if no authentication value is set or if the set value is not valid (for example, the authentication type is not available).
Byte = <CHostDataEx>.ConnLostSecurity	Byte	Returns the numeric representation of the security level that is set for handling an abnormal end of session. The following values are valid: ■ 1: Log off user ■ 2: Restart host computer ■ 3: Lock computer
Byte = <CHostDataEx>.CallbkDelay	Byte	Returns the number of seconds to wait before the host modem calls back the remote.

Table 2-67 CHostDataEx properties and return values

Property	Return value	Description
Bool = <CHostDataEx>.EndSessHostOpts	Bool	Returns whether the host waits for another connection or is cancelled after a normal end of session. The following values are valid: ■ FALSE: Wait for next connection ■ TRUE: Cancel host
Bool = <CHostDataEx>.EnableEndSessSecurity	Bool	Returns whether the end of session security options are enabled for a normal end of session.
Byte = <CHostDataEx>.EndSessSecurity	Byte	Returns the security option that is set for a normal end of session. The following values are valid: ■ 1: Log off user ■ 2: Restart host computer ■ 3: Lock computer
Bool = <CHostDataEx>.BlankHost	Bool	Returns the screen blanking option that is set.
Bool = <CHostDataEx>.AllowRemoteMouse	Bool	Returns whether the remote user has keyboard and mouse control during a session.
Bool = <CHostDataEx>.RebootOnDisconnect	Bool	Returns whether the host computer is restarted after the session ends.
Bool = <CHostDataEx>.PasswordAfterDisc	Bool	Returns whether the host user is logged off after the session ends.
Bool = <CHostDataEx>.LogFailures	Bool	Returns whether the log password failures option is set.
Bool = <CHostDataEx>.AllowDriveSecurity	Bool	Returns whether the drive security option is enabled.
Bool = <CHostDataEx>.UseDirectoryServices	Bool	Returns whether the directory services option is enabled.
String = <CHostDataEx>.DirectoryServiceEntry	String	Returns the directory service settings.

Visual Basic sample code for host functionality

The following Visual Basic sample code retrieves a host data object and modifies 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 the host file in a directory as follows:

```
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

The Awrem32 functions provide the parameters and return values for handling connections between a host and remote computer.

awConnect(FileName)

Creates the connection to the host computer.

[Table 2-68](#) defines the awConnect parameter.

Table 2-68 awConnect parameter

Parameter	Description
Name as string	The fully qualified .chf file name that contains information about the host computer

[Table 2-69](#) defines the awConnect return value.

Table 2-69 awConnect return value

Return value	Description
Boolean	TRUE if command executed

awDisconnect()

Disconnects the host computer. [Table 2-70](#) defines the return value.

Table 2-70 awDisconnect() return value

Return value	Description
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 wildcard characters.

[Table 2-71](#) defines the FileXferFromHost parameters.

Table 2-71 FileXferFromHost parameters

Parameter	Description
HostFile as string	Contains the fully qualified path and file name to be copied from the host computer.
RemoteFile as string	Contains the fully qualified destination path and file name. The HostFile and RemoteFile strings do not have to be identical.

[Table 2-72](#) defines the FileXferFromHost return value.

Table 2-72 FileXferFromHost return value

Return value	Description
Boolean	TRUE if command executed

FileXferToHost(HostFile, RemoteFile)

Copies a file from the remote computer to the host computer. The parameters can contain wildcard characters.

[Table 2-73](#) defines the FileXferToHost parameters.

Table 2-73 FileXferToHost parameters

Parameter	Description
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 from the remote computer. The HostFile and RemoteFile strings do not have to be identical.

[Table 2-74](#) defines the FileXferToHost return value.

Table 2-74 FileXferToHost return value

Return value	Description
Boolean	TRUE if command executed

CreateFolderOnHost(FolderName)

Creates a new folder on the host computer. This function creates a temporary folder on the remote computer, then copies that folder to the host computer.

[Table 2-75](#) defines the CreateFolderOnHost parameter.

Table 2-75 CreateFolderOnHost parameter

Parameter	Description
FolderName as string	Contains the drive and path to create the folder on the host computer

[Table 2-76](#) defines the CreateFolderOnHost return values.

Table 2-76 CreateFolderOnHost return values

Return value	Description
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, this function does not open Microsoft Word if you execute a .doc file.

Table 2-77 defines the ExecuteHostFile parameter.

Table 2-77 ExecuteHostFile parameter

Parameter	Description
FileName as string	Contains the fully qualified path to the file on the host computer

Table 2-78 defines the ExecuteHostFile return value.

Table 2-78 ExecuteHostFile return value

Return value	Description
Boolean	TRUE if command executed

GetError()

Returns the last error as a string. Table 2-79 defines the return value.

Table 2-79 GetError() return value

Return value	Description
String	Returns the last error generated in Awrem32

ConnectionStatus()

Returns the current status of your connection to the host computer. Table 2-80 defines the return value.

Table 2-80 ConnectionStatus() return value

Return value	Description
Short	The possible values include the following: ■ -1 = Lost connection ■ 0 = No connection ■ 1 = Session connected

Visual C++ object definitions

This chapter includes the following topics:

- [About Visual C++ objects](#)
- [CRemoteDataManager methods](#)
- [CRemoteData object](#)
- [CRemoteDataEx object](#)
- [CHostDataManager methods](#)
- [CHostData object](#)
- [CHostDataEx object](#)
- [Awrem32 functions](#)

About Visual C++ objects

The pcAnywhere Automation Server provides the following components to support OLE Automation:

- Winawsvr: Provides the information needed to connect to the pcAnywhere Automation Server and access its interfaces
- Awrem32: Provides the information needed to control pcAnywhere connections

The objects that are described in [Table 3-1](#) comprise Winawsvr.

Table 3-1 Winawsvr objects

Object	Description	Reference
CRemoteDataManager	Provides the methods for creating, opening, modifying, saving, and deleting CRemoteData objects	See “CRemoteDataManager methods” on page 67.
CRemoteData	Defines the parameters for accessing and controlling pcAnywhere remote functionality	See “CRemoteData object” on page 73 and “CRemoteDataEx object” on page 82.
CHostDataManager	Provides the methods for creating, opening, modifying, saving, and deleting CHostData objects	See “CHostDataManager methods” on page 83.
CHostData	Defines the parameters for accessing and controlling pcAnywhere host functionality	See “CHostData object” on page 89 and “CHostDataEx object” on page 99.

Awrem32 has one object, which consists of eight interfaces to support remote control and file transfer sessions.

See [“Awrem32 functions”](#) on page 103.

Some functions, including the Gateway feature, are no longer supported. However, object definitions are provided for use with earlier versions.

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

CRemoteDataManager methods

The CRemoteDataManager methods provide the parameters and return values for accessing and controlling CRemoteData objects.

BSTR CurrentDirectory();

Gets the full path name of the current directory in which pcAnywhere remote objects are stored. [Table 3-2](#) defines the parameter.

Table 3-2 BSTR CurrentDirectory(); parameter

Parameter	Description
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. [Table 3-3](#) defines the parameter.

Table 3-3 BOOL ChangeDirectory parameter

Parameter	Description
LPCTSTR lpszNewDirectory	Name of an existing directory

[Table 3-4](#) defines the return value.

Table 3-4 BOOL ChangeDirectory return value

Return value	Description
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. [Table 3-5](#) defines the parameters.

Table 3-5 BOOL FindFirst parameters

Parameter	Description
LPCTSTR lpszPattern	File name pattern to filter object files (an asterisk [*] finds all files in the current directory)
BSTR FAR * pbstrFullQualName	Return buffer for the full path name of the remote object file that matches the specified pattern

[Table 3-6](#) defines the return value.

Table 3-6 BOOL FindFirst return value

Return value	Description
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 that matches the pattern, if any. [Table 3-7](#) defines the parameter.

Table 3-7 BOOL FindNext parameter

Parameter	Description
BSTR FAR * pbstrFullQualName	Return buffer for the full path name of the remote object file that matches the pattern specified in the original call to FindFirst()

[Table 3-8](#) defines the return value.

Table 3-8 BOOL FindNext return value

Return value	Description
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. [Table 3-9](#) defines the parameters.

Table 3-9 LPDISPATCH Retrieve Object parameters

Parameter	Description
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. [Table 3-10](#) defines the parameters.

Table 3-10 LPDISPATCH RetrieveObjectEx parameters

Parameter	Description
LPCTSTR lpszFQName	The fully qualified remote object file name to be loaded.

Table 3-10 LPDISPATCH RetrieveObjectEx parameters

Parameter	Description
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.

Table 3-11 defines the return value.

Table 3-11 LPDISPATCH RetrieveObjectEx return value

Return value	Description
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CRemoteDataEx object. For an example of how to attach this pointer to a CRemoteDataEx object, see “Visual C++ sample code for remote functionality” on page 82.

LPDISPATCH CreateObject(LPCTSTR lpszFQName);

Creates a CRemoteData object and returns an LPDISPATCH pointer to it. Table 3-12 defines the parameter.

Table 3-12 LPDISPATCH CreateObject parameter

Parameter	Description
LPCTSTR lpszFQName	The fully qualified remote object file name for the new object

[Table 3-13](#) defines the return value.

Table 3-13 LPDISPATCH CreateObject return value

Return value	Description
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CRemoteData object. For an example of how to attach this pointer to a CRemoteData object, see “Visual C++ sample code for remote functionality” on page 82.

LPDISPATCH CreateObjectEx(LPCTSTR lpszFQName);

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

[Table 3-14](#) defines the parameter.

Table 3-14 LPDISPATCH CreateObjectEx parameter

Parameter	Description
LPCTSTR lpszFQName	The fully qualified remote object file name for the new object

[Table 3-15](#) defines the return value.

Table 3-15 LPDISPATCH CreateObjectEx return value

Return value	Description
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CRemoteDataEx object. For an example of how to attach this pointer to a CRemoteDataEx object, see “Visual C++ sample code for remote functionality” on page 82.

BOOL DeleteObject(LPCTSTR lpszFQName, LPCTSTR lpszPassword);

Deletes a remote object file. [Table 3-16](#) defines the parameters.

Table 3-16 BOOL DeleteObject parameters

Parameter	Description
LPCTSTR lpszFQName	The fully qualified remote object file name of the object to be deleted.
LPCTSTR lpszPassword	Object password. May be NULL.

[Table 3-17](#) defines the return value.

Table 3-17 BOOL DeleteObject return value

Return value	Description
BOOL	TRUE if object is deleted

BOOL Launch(LPCTSTR lpszFQName);

Launches a remote object file, which opens the pcAnywhere remote terminal window. [Table 3-18](#) defines the parameter.

Table 3-18 BOOL Launch parameter

Parameter	Description
LPCTSTR lpszFQName	The fully qualified remote object file of object to be launched

[Table 3-19](#) defines the return value.

Table 3-19 BOOL Launch return values

Return value	Description
BOOL	TRUE if object is successfully launched

CRemoteData object

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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 GetAutoLoginName();  
void SetAutoLoginName(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 GetAutoLoginPassword();  
void SetAutoLoginPassword(LPCTSTR lpszNewValue);
```

The password for this object.

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

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

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

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

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

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

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

Controls whether sessions using this object are logged.

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

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

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

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

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

Sets the connection types.

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

The following connection types are available:

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

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

Remote object Detail methods

When a remote object is assigned a connection type, the device details are set to valid default values. The following connection types have advanced configuration options that can be set in your application:

- COM devices
- Network (TCP/IP, SPX) gateway devices
- NetBIOS devices
- ISDN via CAPI 2.0 (European ISDN only) devices

COM device details

Sets the communications parity level.

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

Communications parity values include:

- None
- Odd
- Even
- Mark
- Space

Sets the flow control level.

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

Flow control values include:

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

Sets the connection start setting.

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

Connection start values include:

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

Sets the connection end values.

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

Connection end values include:

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

Sets the connection speed.

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

Connection speed values include:

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

Network (TCP/IP, SPX) device details for Gateways

The following properties can be used only with pcAnywhere 9.2x connections.

Connect through a pcAnywhere Gateway (TRUE).

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

Name of the pcAnywhere Gateway to use.

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

Class of the pcAnywhere Gateway to use.

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

Parity value of the pcAnywhere Gateway to use.

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

Parity values include:

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

NetBIOS device details

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

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

ISDN via CAPI 2.0 device details

Activates channel bonding (uses two ISDN channels for one connection) if TRUE.

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

Sets any additional CAPI extensions that are needed for communications.

```
BSTR GetCapiExtensions();

void SetCapiExtensions(LPCTSTR lpszNewValue);
```

Remote object methods

Following are the normal methods of the remote object (they are not used to get and set properties):

- [short ConnectionTypes\(\);](#)
- [BSTR FirstConnectionType\(\);](#) and [BSTR NextConnectionType\(\);](#)
- [BOOL FindConnectionType\(LPCTSTR lpszConnectionType\);](#)
- [short CountryCodes\(\);](#)
- [BSTR FirstCountryCode\(\);](#) and [BSTR NextCountryCode\(\);](#)
- [BOOL ReadObject\(LPCTSTR lpszPassword\);](#)
- [BOOL WriteObject\(LPCTSTR lpszPassword\);](#)

short ConnectionTypes();

Returns the number of connection types available.

Table 3-20 short ConnectionTypes(); return value

Return value	Description
Short	The number of connection types available on this computer

BSTR FirstConnectionType(); and BSTR NextConnectionType();

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

Table 3-21 BSTR FirstConnectionType(); and BSTR NextConnectionType(); return value

Return value	Description
BSTR	The name of a supported connection device type

BOOL FindConnectionType(LPCTSTR lpszConnectionType);

Returns TRUE if the connection type that is passed in exists on the computer. [Table 3-21](#) defines the parameter.

Table 3-22 BOOL FindConnectionType parameter

Parameter	Description
LPCTSTR lpszConnectionType	The name of a connection device type

[Table 3-22](#) defines the return value.

Table 3-23 BOOL FindConnectionType return value

Return value	Description
BOOL	TRUE if this device type is available

short CountryCodes();

Returns the number of country codes available.

Table 3-24 short CountryCodes(); return value

Return value	Description
Short	The number of country codes available

BSTR FirstCountryCode(); and BSTR NextCountryCode();

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

Table 3-25 BSTR FirstCountryCode(); and BSTR NextCountryCode(); return value

Return value	Description
BSTR	The first or next country code string

BOOL ReadObject(LPCTSTR lpszPassword);

Reads the object data from the remote object file. [Table 3-26](#) defines the parameter.

Table 3-26 BOOL ReadObject parameter

Parameter	Description
LPCTSTR lpszPassword	The object password

[Table 3-27](#) defines the return value.

Table 3-27 BOOL ReadObject return value

Return value	Description
BOOL	TRUE if object is successfully read

BOOL WriteObject(LPCTSTR lpszPassword);

Writes the object data out to the remote object file. [Table 3-28](#) defines the parameter.

Table 3-28 BOOL WriteObject parameter

Parameter	Description
LPCTSTR lpszPassword	The object password

[Table 3-29](#) defines the return value.

Table 3-29 BOOL WriteObject return value

Return value	Description
BOOL	TRUE if object is successfully written

CRemoteDataEx object

The CRemoteDataEx object contains the same functionality as the CRemoteData object with the following additional Get and 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 DenyLowerEncrypt value
void SetDenyLowerEncrypt(BOOL bNewValue);
BSTR GetAutoDomain(); //Returns the AutoDomain value
void SetAutoDomain(LPCTSTR lpszNewValue);
```

Visual C++ sample code for remote functionality

The following 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

The CHostDataManager methods provide the parameters and return values for accessing and controlling CHostData objects.

BSTR CurrentDirectory();

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

Table 3-30 BSTR CurrentDirectory(); return value

Return value	Description
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. [Table 3-31](#) defines the parameter.

Table 3-31 BOOL ChangeDirectory parameter

Parameter	Description
LPCTSTR lpszNewDirectory	Name of an existing directory

[Table 3-32](#) defines the return value.

Table 3-32 BOOL ChangeDirectory return value

Return value	Description
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. [Table 3-33](#) defines the parameters.

Table 3-33 BOOL FindFirst parameters

Parameter	Description
LPCTSTR lpszPattern	File name pattern to filter object files (an asterisk [*] finds all host files in the current directory)
BSTR FAR * pbstrFullQualName	Return buffer for the full path name of the remote object file that matches the specified pattern

[Table 3-34](#) defines the return value.

Table 3-34 BOOL FindFirst return value

Return value	Description
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 that matches the pattern, if any. [Table 3-35](#) defines the parameter.

Table 3-35 BOOL FindNext parameter

Parameter	Description
BSTR FAR * pbstrFullQualName	Return buffer for the full path name of the host object file that matches the pattern specified in the original call to FindFirst()

[Table 3-36](#) defines the return value.

Table 3-36 BOOL FindNext return value

Return value	Description
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. [Table 3-37](#) defines the parameters.

Table 3-37 LPDISPATCH RetrieveObject parameters

Parameter	Description
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.

Table 3-38 defines the return value.

Table 3-38 LPDISPATCH RetrieveObject return value

Return value	Description
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CHostData object. For an example of how to attach this pointer to a CHostData object, see “Visual C++ sample code for host functionality” on page 102.

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

Retrieves a CHostDataEx object by file name. Table 3-39 defines the parameters.

Table 3-39 LPDISPATCH RetrieveObjectEx parameters

Parameter	Description
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.

Table 3-40 defines the return value.

Table 3-40 LPDISPATCH RetrieveObjectEx return value

Return value	Description
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CHostDataEx object. For an example of how to attach this pointer to a CHostDataEx object, see “Visual C++ sample code for host functionality” on page 102.

LPDISPATCH CreateObject(LPCTSTR lpszName);

Creates a CHostData object and returns an LPDISPATCH pointer to it. [Table 3-41](#) defines the parameter.

Table 3-41 LPDISPATCH CreateObject parameter

Parameter	Description
LPCTSTR lpszFQName	The fully qualified host object file name for the new object

[Table 3-42](#) defines the return value.

Table 3-42 LPDISPATCH CreateObject return value

Return value	Description
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CHostData object. For an example of how to attach this pointer to a CHostData object, see “Visual C++ sample code for host functionality” on page 102.

LPDISPATCH CreateObjectEx(LPCTSTR lpszName);

Creates a CHostDataEx object and returns an LPDISPATCH pointer to it. [Table 3-43](#) defines the parameter.

Table 3-43 LPDISPATCH CreateObjectEx parameter

Parameter	Description
LPCTSTR lpszFQName	The fully qualified host object file name for the new object

[Table 3-44](#) defines the return value.

Table 3-44 LPDISPATCH CreateObjectEx return value

Return value	Description
LPDISPATCH	Pointer to an OLE dispatch object. The object is a CHostDataEx object. For an example of how to attach this pointer to a CHostDataEx object, see “Visual C++ sample code for host functionality” on page 102.

BOOL DeleteObject(LPCTSTR lpszFQName, LPCTSTR lpszPassword);

Deletes a host object file. [Table 3-45](#) defines the parameters.

Table 3-45 BOOL DeleteObject parameters

Parameter	Description
LPCTSTR lpszFQName	The fully qualified host object file name of the object to be deleted.
LPCTSTR lpszPassword	Object password. May be NULL.

[Table 3-46](#) defines the return value.

Table 3-46 BOOL DeleteObject return value

Return value	Description
BOOL	TRUE if object is deleted

BOOL Launch(LPCTSTR lpszFQName);

Launches a host object file, which opens the pcAnywhere host terminal window. [Table 3-47](#) defines the parameter.

Table 3-47 BOOL Launch parameter

Parameter	Description
LPCTSTR lpszFQName	The fully qualified host object file of object to be launched

[Table 3-48](#) defines the return value.

Table 3-48 BOOL Launch return value

Return value	Description
BOOL	TRUE if object is successfully launched

CHostData object

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.

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

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

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

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

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

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

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

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

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

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

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

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

Controls whether sessions using this object are logged.

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

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

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

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

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

Host object Detail methods

When a host object is assigned a connection type, the device details are set to valid default values. The following connection types have advanced configuration options that can be set in your application:

- COM devices
- Network (TCP/IP, SPX) gateway devices
- NetBIOS devices
- NASI/NCSI devices
- ISDN via CAPI 2.0 (European ISDN only) devices

COM device details

The following code 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:

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

If the requested connection type is already on 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 connection type that is passed in 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 support up to two assigned connection types. The AssignConnection method returns FALSE if it detects an attempt to exceed this limit.

The following connection types are available:

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

The name of a TAPI device can also 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()` and `NextConnectionType()` to search for available devices.

The following code 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.

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

The following code sets the communications parity levels:

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

Communication parity values include:

- None
- Odd
- Even
- Mark
- Space

The following code sets the flow control levels:

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

Flow control values include:

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

The following code sets the connection start values:

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

Connection start values include:

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

The following code sets the connection end values:

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

Connection end values include:

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

The following code sets the connection speed:

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

Connection speed values include:

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

Network (TCP/IP, SPX) device details for Gateways

The following properties can be used only with pcAnywhere 9.2x connections.

Connect through a pcAnywhere Gateway (TRUE).

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

Name of the pcAnywhere Gateway to use.

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

Class of the pcAnywhere Gateway to use.

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

Parity values of the pcAnywhere Gateway to use.

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

Parity values include:

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

NetBIOS device details

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

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

NASI/NCSI device details

Sets the user name for the NASI server.

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

Sets the user password for NASI server.

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

Sets the NASI session name.

```
BSTR GetNasiSessionName();  
void SetNasiSessionName(LPCTSTR lpszNewValue);  
BOOL GetNasiSessionNameAvailable();  
void SetNasiSessionNameAvailable(BOOL bNewValue);
```

Specifies the NASI server to use.

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

Specifies the NASI Service to use.

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

Specifies the NASI Port to use.

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

ISDN via CAPI 2.0 device details

Activates channel bonding (uses two ISDN channels for one connection) if TRUE.

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

Sets any additional CAPI extensions that are needed for communications.

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

Host object methods

Following are the normal methods of the object (they are not used to get and set properties):

- [short ConnectionTypes\(\);](#)
- [BSTR FirstConnectionType\(\);](#) and [BSTR NextConnectionType\(\);](#)
- [BOOL FindConnectionType\(LPCTSTR lpszConnectionType\);](#)
- [short MaxAssignedConnections\(\)](#)
- [short AssignedConnections\(\)](#)
- [BSTR FirstAssignedConnection\(\);](#) and [BSTR NextAssignedConnection \(\);](#)

- `BOOL FindAssignedConnection (LPCTSTR lpszConnectionType);`
- `short CountryCodes();`
- `BSTR FirstCountryCode();` and `BSTR NextCountryCode();`
- `BOOL ReadObject(LPCTSTR lpszPassword);`
- `BOOL WriteObject(LPCTSTR lpszPassword);`

short ConnectionTypes();

Returns the number of connection types available.

Table 3-49 short Connection Types(); return value

Return value	Description
Short	The number of connection types available on this computer

BSTR FirstConnectionType(); and BSTR NextConnectionType();

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 connection types can be used with the SetConnectionType() function.

Table 3-50 BSTR FirstConnectionType(); and BSTR NextConnectionType(); return value

Return value	Description
BSTR	The name of a supported connection device type

BOOL FindConnectionType(LPCTSTR lpszConnectionType);

Returns TRUE if the connection type that is passed in exists on the computer. [Table 3-51](#) defines the parameter.

Table 3-51 BOOL FindConnectionType parameter

Parameter	Description
LPCTSTR lpszConnectionType	The name of a connection device type

Table 3-52 defines the return value.

Table 3-52 BOOL FindConnectionType return value

Return value	Description
BOOL	TRUE if this device type is available

short MaxAssignedConnections()

Returns the maximum number of connection types that can be assigned at the same time (currently two).

Table 3-53 short MaxAssignedConnections() return value

Return value	Description
Short	The maximum number of connection type assignments

short AssignedConnections()

Returns the number of assigned connection types.

Table 3-54 short AssignedConnections() return value

Return value	Description
Short	The number of assigned connection types on this computer

BSTR FirstAssignedConnection(); and BSTR NextAssignedConnection ();

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

Table 3-55 BSTR FirstAssignedConnection(); and BSTR NextAssignedConnection(); return value

Return value	Description
BSTR	The name of a supported connection device type

**BOOL FindAssignedConnection (LPCTSTR
lpszConnectionType);**

Returns TRUE if the connection type that is passed in is currently assigned on the computer. [Table 3-56](#) defines the parameter.

Table 3-56 BOOL FindAssignedConnection parameter

Parameter	Description
LPCTSTR lpszConnectionType	The name of a connection device type

[Table 3-57](#) defines the return value.

Table 3-57 BOOL FindAssignedConnection return value

Return value	Description
BOOL	TRUE if this device type is currently assigned

short CountryCodes();

Returns the number of country codes available.

Table 3-58 short CountryCodes(); return value

Return value	Description
Short	The number of country codes available

BSTR FirstCountryCode(); and BSTR NextCountryCode();

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

Table 3-59 BSTR FirstCountryCode(); and BSTR NextCountryCode(); return value

Return value	Description
BSTR	The first or next country code string

BOOL ReadObject(LPCTSTR lpszPassword);

Reads the object data from the host object file. [Table 3-60](#) defines the parameter.

Table 3-60 BOOL ReadObject parameter

Parameter	Description
LPCTSTR lpszPassword	The object password

[Table 3-61](#) defines the return value.

Table 3-61 BOOL ReadObject return value

Return value	Description
BOOL	TRUE if object is successfully read

BOOL WriteObject(LPCTSTR lpszPassword);

Writes the object data out to the host object file. [Table 3-62](#) defines the parameter.

Table 3-62 BOOL WriteObject parameter

Parameter	Description
LPCTSTR lpszPassword	The object password

[Table 3-63](#) defines the return value.

Table 3-63 BOOL WriteObject return value

Return value	Description
BOOL	TRUE if object is successfully written

CHostDataEx object

The CHostDataEx object contains the same functionality as the CHostData object with the following additional Get and Set methods:

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

CHostDataEx object

```
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);
```

Visual C++ sample code for host functionality

The following sample Visual C++ function creates a host object, sets its connection type to TCP/IP, sets the computer name to the TCP/IP address that is 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");

    // 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

The Awrem32 functions provide the parameters and return values for handling connections between a host and remote computer.

boolean awConnect(BSTR FileName);

Creates the connection to the host computer. [Table 3-64](#) defines the parameter.

Table 3-64 boolean awConnect parameter

Parameter	Description
Name as string	The fully qualified .chf file name that contains information about the host computer

[Table 3-65](#) defines the return value.

Table 3-65 boolean awConnect return value

Return value	Description
Boolean	Executes the command

boolean awDisconnect();

Disconnects the host computer.

Table 3-66 boolean awDisconnect(); return value

Return value	Description
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 wildcard characters.

Table 3-67 defines the parameters.

Table 3-67 boolean FileXferFromHost parameters

Parameter	Description
HostFile as string	Contains the fully qualified path and file name to be copied from the host computer.
RemoteFile as string	Contains the fully qualified destination path and file name. The HostFile and RemoteFile strings do not have to be identical.

Table 3-68 defines the return value.

Table 3-68 boolean FileXferFromHost return value

Return value	Description
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 wildcard characters.

Table 3-69 defines the parameters.

Table 3-69 boolean FileXferToHost parameters

Parameter	Description
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 from the remote computer. The HostFile and RemoteFile strings do not have to be identical.

[Table 3-70](#) defines the return value.

Table 3-70 boolean FileXferToHost return value

Return value	Description
Boolean	TRUE if command executed

boolean CreateFolderOnHost(BSTR FolderName);

Creates a new folder on the host computer. This function creates a temporary folder on the remote computer, then copies that folder to the host computer.

[Table 3-71](#) defines the parameter.

Table 3-71 boolean CreateFolderOnHost parameter

Parameter	Description
FolderName as string	Contains the drive and path to create the folder on the host computer

[Table 3-72](#) defines the return value.

Table 3-72 boolean CreateFolderOnHost return value

Return value	Description
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.

[Table 3-73](#) defines the parameter.

Table 3-73 boolean ExecuteHostFile parameter

Parameter	Description
FileName as string	Contains the fully qualified path to the file on the host computer

Table 3-74 defines the return value.

Table 3-74 boolean ExecuteHostFile return value

Return value	Description
Boolean	TRUE if command executed

BSTR GetError();

Returns the last error as a string.

Table 3-75 BSTR GetError(); return value

Return value	Description
String	Returns the last error generated in Awrem32

short ConnectionStatus();

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

Table 3-76 short ConnectionStatus(); return value

Return value	Description
Short	The possible values include the following: ■ -1 = Lost connection ■ 0 = No connection ■ 1 = Session connected

Index

A

- API libraries 12
- automation controllers
 - about Visual Basic 10
 - about Visual C++ 11
- automation server. *See* pcAnywhere Automation Server
- Awrem32
 - functions 61, 103
 - type library 12

C

- C++. *See* Visual C++
- CAPI connections
 - host properties 51, 95
 - remote properties 31, 78
- CHostData
 - using Visual Basic 40
 - using Visual C++ 89
- CHostDataEx
 - using Visual Basic 52
 - using Visual C++ 99
- CHostDataManager
 - using Visual Basic 35
 - using Visual C++ 83
- class definitions
 - importing 12
 - viewing 13
- code samples
 - Visual Basic 33, 60
 - Visual C++ 82, 102
- COM devices
 - host device details 47, 90
 - remote device details 27, 76
- connection types
 - assigning 44
 - host properties 42, 90, 95
 - remote properties 24, 75, 79
 - unassigning 45

- connections
 - ending 62, 103
 - returning errors 106
 - returning status 64, 106
 - starting 40, 61, 72, 103
- country codes
 - on host 98
 - on remote 80
- CRemoteData
 - using Visual Basic 22
 - using Visual C++ 73
- CRemoteDataEx
 - using Visual Basic 32
 - using Visual C++ 82
- CRemoteDataManager
 - using Visual Basic 17
 - using Visual C++ 67

D

- devices. *See* connection types

E

- encryption settings
 - on host 52, 99
 - on remote 32, 33, 82
- errors, returning 64, 106

F

- file transfer
 - from host 62, 104
 - to host 63, 104
- files, running on host 64, 105
- flow control 27, 47, 76, 92
- folders, creating on host 63, 105

G

- Gateway properties 31, 51, 78, 93
- GUIDs 9

H

host objects

- creating 38, 87
- deleting 39, 88
- device details 90
- dialing properties 46
- directories 35, 83
- finding 36-38, 84-86
- methods 95
- passwords 99
- starting 40, 88

I

identifiers. *See* GUIDs

ISDN, CAPI connections

- host properties 51, 95
- remote properties 31, 78

M

modem connections

- on host
 - COM properties 47, 90
 - country codes 98
 - dialing properties 46
 - TAPI devices 91
- on remote
 - COM properties 27, 76
 - country codes 80
 - dialing properties 26
 - TAPI devices 75

N

NASI devices 94

NCSI devices 94

NetBIOS connections

- host properties 50, 94
- remote properties 30, 78

network connections

- on host
 - Gateway properties 51, 93
 - NetBIOS properties 50, 94
- on remote
 - Gateway properties 31, 78
 - NetBIOS properties 30, 78

O

OLE Automation. *See* pcAnywhere Automation Server

P

parity 27, 47, 76, 91

passwords

- on host objects 99
- on remote objects 81

pcAnywhere Automation Server

- about 8
- accessing with Visual Basic 10
- accessing with Visual C++ 11
- example uses 9
- registering GUIDs 9
- type libraries 12

R

remote engine

- automatically registering 9
- manually registering 10

remote objects

- creating 70
- deleting 21, 72
- device details 75
- dialing properties 26
- directories 17, 67
- files 18-20, 36, 68-70
- methods 79
- passwords 81

S

status, returning 64, 106

T

TAPI devices

- on host 91
- on remote 75

type libraries 12

V

Visual Basic

- accessing pcAnywhere Automation Server 10
- Awrem32 functions 61
- CHostData object 40

Visual Basic (*continued*)

- CHostDataEx object 52
- CHostDataManager object 35
- code samples 33, 60
- CRemoteData object 22
- CRemoteDataEx object 32
- CRemoteDataManager object 17

Visual C++

- accessing pcAnywhere Automation Server 11
- Awrem32 functions 103
- CHostData object 89
- CHostDataEx object 99
- CHostDataManager object 83
- code samples 82, 102
- CRemoteData object 73
- CRemoteDataEx object 82
- CRemoteDataManager object 67
- importing classes 12
- including Winawsvr.h 13

W**Winawsvr**

- executable 10
- header file 13
- objects
 - CHostData 40, 89
 - CHostDataEx 52, 99
 - CHostDataManager 35, 83
 - CRemoteData 22, 73
 - CRemoteDataEx 32, 82
 - CRemoteDataManager 17, 67
- type library 12