

OLE Automation Guide

OLE Automation Guide

Legal Notice

Copyright © 2008 Symantec Corporation. All rights reserved.

Symantec, the Symantec Logo are trademarks or registered trademarks of Symantec Corporation or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.

This Symantec product may contain third party software for which Symantec is required to provide attribution to the third party ("Third Party Programs"). Some of the Third Party Programs are available under open source or free software licenses. The License Agreement accompanying the Software does not alter any rights or obligations you may have under those open source or free software licenses. Please see the Third Party Legal Notice Appendix to this Documentation or TPIP ReadMe File accompanying this Symantec product for more information on the Third Party Programs.

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation/reverse engineering. No part of this document may be reproduced in any form by any means without prior written authorization of Symantec Corporation and its licensors, if any.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. SYMANTEC CORPORATION SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

The Licensed Software and Documentation are deemed to be commercial computer software as defined in FAR 12.212 and subject to restricted rights as defined in FAR Section 52.227-19 "Commercial Computer Software - Restricted Rights" and DFARS 227.7202, "Rights in Commercial Computer Software or Commercial Computer Software Documentation", as applicable, and any successor regulations. Any use, modification, reproduction release, performance, display or disclosure of the Licensed Software and Documentation by the U.S. Government shall be solely in accordance with the terms of this Agreement.

Symantec Corporation
20330 Stevens Creek Blvd.
Cupertino, CA 95014

<http://www.symantec.com>

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Technical Support

Symantec Technical Support maintains support centers globally. Technical Support's primary role is to respond to specific queries about product features and functionality. The Technical Support group also creates content for our online Knowledge Base. The Technical Support group works collaboratively with the other functional areas within Symantec to answer your questions in a timely fashion. For example, the Technical Support group works with Product Engineering and Symantec Security Response to provide alerting services and virus definition updates.

Symantec's maintenance offerings include the following:

- A range of support options that give you the flexibility to select the right amount of service for any size organization
- Telephone and Web-based support that provides rapid response and up-to-the-minute information
- Upgrade assurance that delivers automatic software upgrade protection
- Global support that is available 24 hours a day, 7 days a week
- Advanced features, including Account Management Services

For information about Symantec's Maintenance Programs, you can visit our Web site at the following URL:

www.symantec.com/techsupp/

Contacting Technical Support

Customers with a current maintenance agreement may access Technical Support information at the following URL:

www.symantec.com/techsupp/

Before contacting Technical Support, make sure you have satisfied the system requirements that are listed in your product documentation. Also, you should be at the computer on which the problem occurred, in case it is necessary to replicate the problem.

When you contact Technical Support, please have the following information available:

- Product release level
- Hardware information
- Available memory, disk space, and NIC information
- Operating system

- Version and patch level
- Network topology
- Router, gateway, and IP address information
- Problem description:
 - Error messages and log files
 - Troubleshooting that was performed before contacting Symantec
 - Recent software configuration changes and network changes

Licensing and registration

If your Symantec product requires registration or a license key, access our technical support Web page at the following URL:

www.symantec.com/techsupp/

Customer service

Customer service information is available at the following URL:

www.symantec.com/techsupp/

Customer Service is available to assist with the following types of issues:

- Questions regarding product licensing or serialization
- Product registration updates, such as address or name changes
- General product information (features, language availability, local dealers)
- Latest information about product updates and upgrades
- Information about upgrade assurance and maintenance contracts
- Information about the Symantec Buying Programs
- Advice about Symantec's technical support options
- Nontechnical presales questions
- Issues that are related to CD-ROMs or manuals

Maintenance agreement resources

If you want to contact Symantec regarding an existing maintenance agreement, please contact the maintenance agreement administration team for your region as follows:

Asia-Pacific and Japan	contractsadmin@symantec.com
Europe, Middle-East, and Africa	semea@symantec.com
North America and Latin America	supportsolutions@symantec.com

Additional enterprise services

Symantec offers a comprehensive set of services that allow you to maximize your investment in Symantec products and to develop your knowledge, expertise, and global insight, which enable you to manage your business risks proactively.

Enterprise services that are available include the following:

Symantec Early Warning Solutions	These solutions provide early warning of cyber attacks, comprehensive threat analysis, and countermeasures to prevent attacks before they occur.
Managed Security Services	These services remove the burden of managing and monitoring security devices and events, ensuring rapid response to real threats.
Consulting Services	Symantec Consulting Services provide on-site technical expertise from Symantec and its trusted partners. Symantec Consulting Services offer a variety of prepackaged and customizable options that include assessment, design, implementation, monitoring, and management capabilities. Each is focused on establishing and maintaining the integrity and availability of your IT resources.
Educational Services	Educational Services provide a full array of technical training, security education, security certification, and awareness communication programs.

To access more information about Enterprise services, please visit our Web site at the following URL:

www.symantec.com

Select your country or language from the site index.

Contents

Technical Support	4
Chapter 1 Using OLE Automation with Symantec pcAnywhere	9
About OLE Automation	9
About the pcAnywhere Automation Server	10
What you can do with the pcAnywhere Automation Server	10
Before you start	11
Automatically registering the remote engine	11
Manually registering the remote engine	11
Accessing the pcAnywhere Automation Server	12
Accessing the pcAnywhere Automation Server with Visual Basic	12
Accessing the pcAnywhere Automation Server with Visual C++	13
Launching host and remote OLE objects	15
Where to find more information	15
Chapter 2 Visual Basic object definitions	17
About Visual Basic objects	17
CRemoteDataManager methods	18
CRemoteData properties	21
Connection type properties	23
Dialing properties	25
COM device properties	26
NetBIOS device properties	29
ISDN via CAPI 2.0 device properties	30
CRemoteDataEx object	31
Visual Basic sample code for remote functionality	32
CHostDataManager methods	34
CHostData properties	36
Connection type properties	38
AssignConnection(ConnectionType) method	41
UnassignConnection(ConnectionType) method	41
Dialing properties	42

- COM device properties 42
- NetBIOS device properties 45
- ISDN via CAPI 2.0 device properties 46
- CHostDataEx object 47
 - Visual Basic sample code for host functionality 54
- Awrem32 functions 55

Chapter 3 Visual C++ object definitions 59

- About Visual C++ objects 59
- CRemoteDataManager methods 60
- CRemoteData object 63
 - Get and Set methods 63
 - Remote object Detail methods 66
 - Remote object methods 68
- CRemoteDataEx object 70
 - Visual C++ sample code for remote functionality 70
- CHostDataManager methods 71
- CHostData object 75
 - Get and Set methods 75
 - Host object Detail methods 76
 - Host object methods 80
- CHostDataEx object 82
 - Visual C++ sample code for host functionality 84
- Awrem32 functions 85

Index 89

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.

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 and host 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.

You can use the pcAnywhere Automation Server to 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.

See [“About Visual Basic objects”](#) on page 17.

See [“About Visual C++ objects”](#) on page 59.

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.

You can register the GUID entries automatically if you are running the external application on a computer on which Symantec pcAnywhere is installed. Otherwise, you must register the GUID entries manually.

See [“Automatically registering the remote engine”](#) on page 11.

See [“Manually registering the remote engine”](#) on page 11.

Automatically registering the remote engine

Before you connect to another computer for the first time using your OLE client, you must self-register the remote engine. You can do this automatically when you start a remote object in 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 connection item.

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 installation 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 to interact with OLE Automation servers, such as the pcAnywhere Automation Server. You create a Standard Exe project, and 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.

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 do any of the following:
 - Determine the current directory.
 - Change to another directory.
 - Enumerate the associated data object files in the current directory.

- 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 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 do the following:

- Determine or change the current directory.
- Enumerate through the list of data object files in the current directory.
- 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.

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

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**, and 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**, and 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 let you 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 14.

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 you store all host and remote objects in the default data directory. Before you launch a host or remote object that you created using OLE, ensure that the object is located in the pcAnywhere default data directory. The default directory is the All Users folder in Windows XP, and is the ProgramData folder in Vista.

Where to find more information

For more information about OLE automation, see the following references:

- Box, Don. 1998. *Essential COM*. Reading, Mass.: Addison-Wesley.
- 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 18.
CRemoteData	Defines the parameters for accessing and controlling pcAnywhere remote functionality	See “CRemoteData properties” on page 21. See “CRemoteDataEx object” on page 31.
CHostDataManager	Provides the methods for creating, opening, modifying, saving, and deleting CHostData objects	See “CHostDataManager methods” on page 34.
CHostData	Defines the parameters for accessing and controlling pcAnywhere host functionality	See “CHostData properties” on page 36. See “CHostDataEx object” on page 47.

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

See “Awrem32 functions” on page 55.

Some functions are no longer supported. However, object definitions are provided for use with earlier versions of pcAnywhere.

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.

Table 2-2 CRemoteDataManager methods

CRemoteDataManager method	Parameter or return value	Description
CurrentDirectory()	Return Value = String	Returns the full path name of the current directory in which pcAnywhere remote objects are stored
ChangeDirectory(NewDirectory)	Parameter = NewDirectory	Changes the current directory in which pcAnywhere remote objects are stored

Table 2-2 CRemoteDataManager methods (*continued*)

CRemoteDataManager method	Parameter or return value	Description
FindFirst(Pattern)	Parameter = Pattern as string	Finds the first pcAnywhere remote object file (*.chf) in the current directory; file name pattern filters based on the specified file name pattern (an asterisk [*] finds all files in the current directory)
FindFirst(Name string)	Parameter = Name as string	Return buffer for the full path name of the remote object file (*.chf) that matches the specified pattern
FindNext(Name)	Parameter = Name as string	Return buffer for the full path name of the remote object file that matches the pattern that is specified in the original call to FindFirst()
FindNext(Name)	Return value = Boolean	TRUE if another remote object file that matches the pattern that is specified in the call to FindFirst() is found. The full path name of the matching file is stored in Name
RetrieveObject(Name)	Parameter = Name as string	The fully qualified remote object file name to be loaded
RetrieveObject(AccessMode)	Parameter = AccessMode as integer	Specifies how this object is to be used. This relates to the password protection The options are as follows: ■ 0 = Not specified ■ 1 = View only ■ 2 = View and Modify ■ 3 = Execute
RetrieveObject(Password)	Parameter = Password as string	Object password. May be NULL
RetrieveObjectEx(Name)	Parameter = Name as string	The fully qualified remote object file name to be loaded

Table 2-2 CRemoteDataManager methods (*continued*)

CRemoteDataManager method	Parameter or return value	Description
RetrieveObjectEx(AccessMode)	Parameter = AccessMode as integer	Specifies how this object is to be used. This relates to the password protection The options are as follows: ■ 0 = Not specified ■ 1 = View only ■ 2 = View and Modify ■ 3 = Execute
RetrieveObjectEx>Password)	Parameter = Password as string	Object password. May be NULL
RetrieveObjectEx	Return value = Object	CRemoteDataEx object from the specified file
CreateObject(Name)	Parameter = Name as string	Creates a CRemoteData object and returns an LPDISPATCH pointer to it. The parameter is the fully qualified remote object file name for the new object
CreateObject	Return value = Object	CRemoteData
CreateObjectEx(Name)	Parameter = Name as string	Creates a CRemoteDataEx object and returns an LPDISPATCH pointer to it. The parameter is the fully qualified remote object file name for the new object
CreateObjectEx	Return value = Object	CRemoteDataEx
DeleteObject(Name)	Parameter = Name as string	The fully qualified remote object file name of the object to be deleted
DeleteObject>Password)	Parameter = Password as string	Object password
DeleteObject	Return value = Boolean	TRUE if object is deleted

CRemoteData properties

The properties and parameters that are available for the CRemoteData object are shown as follows.

In [Table 2-3](#), replace the information in angle brackets with the actual values.

Table 2-3 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 31.
<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.

Table 2-3 CRemoteData properties and parameters (*continued*)

Property	Parameter	Description
<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.
<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. Use this property to refresh the local data copy of the remote object.
<CRemoteData>.WriteObject(String)	String	Sets the password of the object. Use this property to create the remote object or to write changes that you have made to the remote object.

The properties and return values for CRemoteData are shown in the following table.

In [Table 2-4](#), replace the information in angle brackets with the actual values.

Table 2-4 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

Table 2-4 CRemoteData properties and return values (*continued*)

Property	Return value	Description
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
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

The following table defines the connection type properties and parameters.

In [Table 2-5](#), replace the information in angle brackets with the actual values.

Table 2-5 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 ■ TCP/IP ■ SPX ■ NetBIOS ■ ISDN via CAPI 2.0 ■ Modem name (as it appears on the computer)

The following table defines the connection type properties and return values.
In [Table 2-6](#), replace the information in angle brackets with the actual values.

Table 2-6 Connection type properties and return values

Property	Return value	Description
String = <CRemoteData>.ConnectionTypes	String	Returns the connection type of the remote object.

Table 2-6 Connection type properties and return values (*continued*)

Property	Return value	Description
Integer = <CRemoteData>.ConnectionType	Integer	Returns the number of available connection types. The following are examples of valid connection types: ■ COM1 ■ COM2 ■ COM3 ■ COM4 ■ TCP/IP ■ SPX ■ NetBIOS ■ 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

The following table defines the properties and parameters for setting the dialing properties for modem connections.

In [Table 2-7](#), replace the information in angle brackets with the actual values.

Table 2-7 Properties and parameters for dialing properties

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

Table 2-7 Properties and parameters for dialing properties (continued)

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

The following table defines the properties and return values for modem dialing properties.

In Table 2-8, replace the information in angle brackets with the actual values.

Table 2-8 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

The following table describes the properties and parameters that let you customize the port settings for modem and other COM-based connections.

In Table 2-9, replace the information in angle brackets with the actual values.

Table 2-9 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
<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)

Table 2-9COM device properties and parameters (continued)

Property	Parameter	Description
<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

The following table describes the COM device properties and return values.
In [Table 2-10](#), replace the information in angle brackets with the actual values.

Table 2-10COM 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

Table 2-10 COM device properties and return values (*continued*)

Property	Return value	Description
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)
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

The following table defines the property and parameter for a NetBIOS network device.

In [Table 2-11](#), replace the information in angle brackets with the actual values.

Table 2-11 NetBIOS property and parameter

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

The following table defines the NetBIOS property and return value.

In [Table 2-12](#), replace the information in angle brackets with the actual values.

Table 2-12 NetBIOS property and 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

The following table defines the properties and parameters for European ISDN connections.

In [Table 2-13](#), replace the information in angle brackets with the actual values.

Table 2-13 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

The following table defines the properties and return values for European ISDN connections.

In [Table 2-14](#), replace the information in angle brackets with the actual values.

Table 2-14 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

Table 2-14 ISDN via CAPI 2.0 properties and return values (*continued*)

Property	Return value	Description
String = <CRemoteData>.CapiExtensions	String	Returns the current list of CAPI extensions from the remote object

CRemoteDataEx object

The CRemoteDataEx object contains the same functionality as the CRemoteData object with some additional functionality.

The following table describes the properties and parameters.

In [Table 2-15](#), replace the information in angle brackets with the actual values.

Table 2-15 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.

The following table describes the properties and return values for the CRemoteDataEx object.

In [Table 2-16](#), replace the information in angle brackets with the actual values.

Table 2-16 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.
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 ("WINAWSVR.REMOTEDATAMANAGER")  
MsgBox (RemoteDataManager.CurrentDirectory())  
  
RemoteDataManager.ChangeDirectory ("C:\dev\bin.w32\data")  
MsgBox (RemoteDataManager.CurrentDirectory())  
Set RemoteData = RemoteDataManager.CreateObject("test")  
RemoteData.ConnectionType = "TCP/IP"  
RemoteData.ComputerName = "Host1"  
s = RemoteData.ConnectionType  
MsgBox (s)  
s = RemoteData.ComputerName  
MsgBox (s)  
RemoteData.WriteObject (0)  
End Sub
```

CHostDataManager methods

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

CHostDataManager method	Parameter or return value	Description
CurrentDirectory()	Return value = String	The full path name of the current pcAnywhere data directory in which pcAnywhere host objects are stored
FindFirst(Pattern)	Parameter = Pattern as string	File name pattern to filter object files (an asterisk [*] finds all files in the current directory)
FindFirst(Name string)	Parameter = Name as string	Return buffer for the full path name of the host object file that matches the specified pattern

CHostDataManager method	Parameter or return value	Description
FindFirst	Return value = 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)	Parameter = 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()
FindNext	Return value = Boolean	TRUE if another host object file matching the pattern specified in the call to FindFirst() is found. The full path name of the matching file is stored in Name
RetrieveObject(Name)	Parameter = Name as string	The fully qualified host object file name to be loaded
RetrieveObject(AccessMode)	Parameter = AccessMode as integer	Specifies how this object is to be used. This relates to the password protection. The options are as follows: ■ 0 = Not specified ■ 1 = View only ■ 2 = View and Modify ■ 3 = Execute
RetrieveObject(Password)	Parameter = Password as string	Object password. May be NULL
RetrieveObject	Return value = Object	CHostData object from the specified file
RetrieveObjectEx(Name)	Parameter = Name as string	The fully qualified host object file name to be loaded
RetrieveObjectEx(AccessMode)	Parameter = AccessMode as integer	Specifies how this object is to be used. This relates to the password protection The options are as follows: ■ 0 = Not specified ■ 1 = View only ■ 2 = View and Modify ■ 3 = Execute

CHostDataManager method	Parameter or return value	Description
RetrieveObjectEx(Password)	Parameter = Password as string	Object password. May be NULL
RetrieveObjectEx	Return value = Object	CHostDataEx object from the specified file
CreateObject(Name)	Parameter = Name as string	Creates a CHostData object and returns an LPDISPATCH pointer to it. The parameter is the fully qualified host object file name for the new object
CreateObject	Return value = Object	CHostData
CreateObjectEx(Name)	Parameter = Name as string	Creates a CHostDataEx object and returns an LPDISPATCH pointer to it. The parameter is the fully qualified host object file name for the new object
CreateObjectEx	Return value = Object	CHostDataEx
DeleteObject(Name)	Parameter = Name as string	The fully qualified host object file name of the object to be deleted
DeleteObject(Password)	Parameter = Password as string	Object password
DeleteObject	Return value = Boolean	TRUE if object is deleted
Launch(Name)	Parameter = Name as string	Launches a host object file, which opens the pcAnywhere host terminal window. The parameter is the fully qualified host object file of the object to be launched
Launch	Return value = Boolean	TRUE if object is successfully launched

CHostData properties

The following table describes the properties and parameters that are available for the CHostData object.

In [Table 2-17](#), replace the information in angle brackets with the actual values.

Table 2-17 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.
<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. You can use this property to refresh the local data copy of the host object.
<CHostData>.WriteObject(String)	String	Sets the password for the object. Use this property to create the host object or to write changes that you have made to the host object.

The following table describes the CHostData properties and return values.

In [Table 2-18](#), replace the information in angle brackets with the actual values.

Table 2-18 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
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

The following table describes the connection type properties and parameters.

In [Table 2-19](#), replace the information in angle brackets with the actual values.

Table 2-19 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 ■ TCP/IP ■ SPX ■ NetBIOS ■ 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.
<CHostData>.UnassignConnection(String)	String	Sets the connection type that is defined in the string to inactive status.

The following table describes the connection type properties and return values.
In [Table 2-20](#), replace the information in angle brackets with the actual values.

Table 2-20 Connection type properties and return values

Property	Return value	Description
String = <CHostData>.ConnectionTypes	String	Returns the connection type of the host object.

Table 2-20 Connection type properties and return values (*continued*)

Property	Return value	Description
Integer = <CHostData>.ConnectionType	Integer	Returns the number of available connection types. The following are examples of valid connection types: ■ COM1 ■ COM2 ■ COM3 ■ COM4 ■ TCP/IP ■ SPX ■ NetBIOS ■ 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.
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.

The following table defines the AssignConnection parameter:

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

The following table defines the 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.

The following table defines the UnassignConnection parameter:

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

The following table defines the UnassignConnection return value:

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

Dialing properties

The following table defines the properties and return values for setting the dialing properties for modem connections.

In [Table 2-21](#), replace the information in angle brackets with the actual values.

Table 2-21 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

The following table defines the properties and return values for modem dialing properties.

In [Table 2-22](#), replace the information in angle brackets with the actual values.

Table 2-22 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

The following table defines the properties and parameters that let you customize the port settings for modem and other COM-based connections.

In [Table 2-23](#), replace the information in angle brackets with the actual values.

Table 2-23 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
<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)

Table 2-23COM device properties and parameters (continued)

Property	Parameter	Description
<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

The following table describes the COM device properties and return values.
In [Table 2-24](#), replace the information in angle brackets with the actual values.

Table 2-24COM 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

Table 2-24 COM device properties and return values (*continued*)

Property	Return value	Description
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)
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

The following table defines the property and parameter for NetBIOS network devices.

In [Table 2-25](#), replace the information in angle brackets with the actual values.

Table 2-25 NetBIOS property and parameter

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

The following table defines the property and return value for NetBIOS network devices.

In [Table 2-26](#), replace the information in angle brackets with the actual values.

Table 2-26 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

The following table defines the properties and parameters for European ISDN connections.

In [Table 2-27](#), replace the information in angle brackets with the actual values.

Table 2-27 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 2-28](#) defines the properties and return values for European ISDN connections.

Table 2-28 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

Table 2-28 ISDN via CAPI 2.0 properties and return values (*continued*)

Property	Return value	Description
String = <CHostData>.CapiExtensions	String	Returns the current list of CAPI extensions from the host object

CHostDataEx object

The CHostDataEx object contains the same functionality as the CHostData object with some additional functionality.

The following table describes the properties and parameters.

In [Table 2-29](#), replace the information in angle brackets with the actual values.

Table 2-29 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.

Table 2-29 CHostDataEx properties and parameters (*continued*)

Property	Parameter	Description
<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.
<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

Table 2-29 CHostDataEx properties and parameters (*continued*)

Property	Parameter	Description
<CHostData>.EnableConnLostSecurity(Bool)	Bool	Activates or deactivates the end of session security options for sessions that end abnormally.
<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: 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 2-29 CHostDataEx properties and parameters *(continued)*

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.

The following table describes the properties and return values for the CHostDataEx object.

In [Table 2-30](#), replace the information in angle brackets with the actual values.

Table 2-30 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.

Table 2-30 CHostDataEx properties and return values (*continued*)

Property	Return value	Description
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).
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.

Table 2-30 CHostDataEx properties and return values (*continued*)

Property	Return value	Description
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.
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: 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).

Table 2-30 CHostDataEx properties and return values (*continued*)

Property	Return value	Description
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.
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.

Table 2-30 CHostDataEx properties and return values (continued)

Property	Return value	Description
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 the host and remote computers.

Awrem32 function	Parameter or return value	Description
awConnect(fileName)	Parameter = Name as string	The fully qualified .chf file name that contains information about the host computer
awConnect	Return value = Boolean	TRUE if command executed

Awrem32 function	Parameter or return value	Description
awDisconnect()	Return value = Boolean	After calling this function, the calling program must delete the object (C++ - delete IAwrem32X*, VB - set ObjectName = Nothing;)
FileXferFromHost(HostFile)	Parameter = HostFile as string	Contains the fully qualified path and file name to be copied from the host computer
FileXferFromHost(RemoteFile)	Parameter = RemoteFile as string	Contains the fully qualified destination path and file name. The HostFile and RemoteFile strings do not have to be identical
FileXferFromHost	Return value = Boolean	TRUE if command executed
FileXferToHost(HostFile)	Parameter = HostFile as string	Copies a file from the remote computer to the host computer. The parameters can contain wildcard characters The parameter contains the fully qualified destination path and file name
FileXferToHost(RemoteFile)	Parameter = 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
CreateFolderOnHost(FolderName)	Parameter = FolderName as string	Contains the drive and path to create the folder on the host computer
CreateFolderOnHost	Return value = Boolean	TRUE if command executed
ExecuteHostFile(FileName)	Parameter = FileName as string	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. The parameter contains the fully qualified path to the file on the host computer

Awrem32 function	Parameter or return value	Description
ExecuteHostFile	Return value = Boolean	TRUE if command executed
GetError()	Return value = String	Returns the last error generated in Awrem32
ConnectionStatus()	Return value = Short	Returns the current status of your connection to the host computer 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 60.
CRemoteData	Defines the parameters for accessing and controlling pcAnywhere remote functionality	See “CRemoteData object” on page 63. See “CRemoteDataEx object” on page 70.
CHostDataManager	Provides the methods for creating, opening, modifying, saving, and deleting CHostData objects	See “CHostDataManager methods” on page 71.
CHostData	Defines the parameters for accessing and controlling pcAnywhere host functionality	See “CRemoteData object” on page 63. See “CRemoteDataEx object” on page 70.

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

See “Awrem32 functions” on page 85.

Some functions 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.

Method	Parameter or return value	Description
BSTR CurrentDirectory();	Parameter = BSTR	The full path name of the current pcAnywhere data directory in which pcAnywhere remote objects are stored
BOOL ChangeDirectory(LPCTSTR lpszNewDirectory);	Parameter = LPCTSTR lpszNewDirectory Return value = BOOL	Changes the current directory in which pcAnywhere remote objects are stored. Parameter is the name of an existing directory Return value is TRUE if successful

Method	Parameter or return value	Description
BOOL FindFirst(LPCTSTR lpszPattern, BSTR FAR*);	Parameter = LPCTSTR lpszPattern	File name pattern to filter object files (an asterisk [*] finds all files in the current directory)
BOOL FindFirst(pbstrFullQualName);	Parameter = BSTR FAR * pbstrFullQualName	Return buffer for the full path name of the remote object file that matches the specified pattern
BOOL FindFirst	Return value = BOOL	TRUE if a remote object file matching the specified pattern is found. The full path name of the matching file is stored in pbstrFullQualName
BOOL FindNext(BSTR FAR* pbstrFullQualName);	Parameter = BSTR FAR * pbstrFullQualName Return value = BOOL	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 Return value is 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);	Parameter = LPCTSTR lpszFQName	Retrieves a CRemoteData object by file name Parameter is the fully qualified remote object file name to be loaded
LPDISPATCH RetrieveObject(short wAccessMode);	Parameter = short wAccessMode	Specifies how this object is to be used. This relates to the password protection. The options are as follows: ■ 0 = Not specified ■ 1 = View only ■ 2 = View and Modify ■ 3 = Execute
LPDISPATCH RetrieveObject(LPCTSTR lpszPassword);	Parameter = LPCTSTR lpszPassword	Object password. May be NULL

Method	Parameter or return value	Description
LPDISPATCH RetrieveObjectEx(LPCTSTR lpszFQName);	Parameter = LPCTSTR lpszFQName	Retrieves a CRemoteDataEx object by file name Parameter is the fully qualified remote object file name to be loaded
LPDISPATCH RetrieveObjectEx(short wAccessMode);	Parameter = short wAccessMode	Specifies how this object is to be used. This relates to the password protection. The options are as follows: ■ 0 = Not specified ■ 1 = View only ■ 2 = View and Modify ■ 3 = Execute
LPDISPATCH RetrieveObjectEx(LPCTSTR lpszPassword);	Parameter = LPCTSTR lpszPassword	Object password. May be NULL
LPDISPATCH RetrieveObjectEx	Return value = LPDISPATCH	Pointer to an OLE dispatch object. The object is a CRemoteDataEx object. See Visual C++ sample code for remote functionality .
LPDISPATCH CreateObject(LPCTSTR lpszFQName);	Parameter = LPCTSTR lpszFQName Return value = LPDISPATCH	Creates a CRemoteData object and returns an LPDISPATCH pointer to it. Parameter is the fully qualified remote object file name for the new object Return value is a pointer to an OLE dispatch object. The object is a CRemoteData object. See Visual C++ sample code for remote functionality .

Method	Parameter or return value	Description
LPDISPATCH CreateObjectEx(LPCTSTR lpszFQName);	Parameter = LPCTSTR lpszFQName Return value = LPDISPATCH	Creates a CRemoteDataEx object and returns an LPDISPATCH pointer to it Parameter is the fully qualified remote object file name for the new object Return value is a pointer to an OLE dispatch object. The object is a CRemoteDataEx object. See Visual C++ sample code for remote functionality.
BOOL DeleteObject(LPCTSTR lpszFQName);	Parameter = LPCTSTR lpszFQName	Deletes a remote object file. Parameter is the fully qualified remote object file name of the object to be deleted
BOOL DeleteObject(LPCTSTR lpszPassword);	Parameter = LPCTSTR lpszPassword	Object password. May be NULL
BOOL DeleteObject	BOOL	TRUE if object is deleted
BOOL Launch(LPCTSTR lpszFQName);	Parameter = LPCTSTR lpszFQName Return value = BOOL	Launches a remote object file, which opens the pcAnywhere remote terminal window. Parameter is the fully qualified remote object file of object to be launched Return value is 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 whether 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
- ISDN via CAPI 2.0

- 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
- 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 are as follows:

- None
- Odd
- Even
- Mark
- Space

Sets the flow control level.

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

Flow control values are as follows:

- <None>

- XONXOFF
- RTS/CTS
- BOTH

Sets the connection start setting.

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

Connection start values are as follows:

- 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 are as follows:

- 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 are as follows:

- 110
- 300
- 600

- 1200
- 2400
- 4800
- 9600
- 19200
- 38400
- 57600
- 115200

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

The following table describes the normal methods of the remote object (they are not used to get and set properties):

Method	Parameter or return value	Description
short ConnectionTypes();	Return value = Short	The number of connection types available on this computer

Method	Parameter or return value	Description
BSTR FirstConnectionType(); and BSTR NextConnectionType();	Return value = BSTR	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. You can use these returned connection types with the SetConnectionType() function. The return value is the name of a supported connection device type
BOOL FindConnectionType(LPCTSTR lpszConnectionType);	Parameter = LPCTSTR lpszConnectionType Return value = BOOL	The name of a connection device type The return value is TRUE if the connection type that is passed in exists on the computer
short CountryCodes();	Return value = Short	Returns the number of country codes available
BSTR FirstCountryCode(); and BSTR NextCountryCode();	Return value = BSTR	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. You can use these returned values with the SetCountryCode() function The return value is the first or next country code string
BOOL ReadObject(LPCTSTR lpszPassword);	Parameter = LPCTSTR lpszPassword Return value = BOOL	Reads the object data from the remote object file The parameter is the object password The return value is TRUE if object is successfully read
BOOL WriteObject(LPCTSTR lpszPassword);	Parameter = LPCTSTR lpszPassword Return value = BOOL	Writes the object data out to the remote object file. The parameter is the object password The return value is 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, and then launches the remote object:

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

    CRemoteDataManager remoteDM;

    CRemoteData remoteData;

    // First, create the CRemoteDataManager
    remoteDM.CreateDispatch( _T( "WINAHSV.R.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 following table describes the CHostDataManager methods, including the parameters and return values for accessing and controlling CHostData objects:

Method	Parameter or return value	Description
BSTR CurrentDirectory();	Return value = BSTR	Returns the full path name of the current directory in which pcAnywhere host objects are stored
BOOL ChangeDirectory(LPCTSTR lpszNewDirectory);	Parameter = LPCTSTR lpszNewDirectory Return value = BOOL	Changes the current directory in which pcAnywhere host objects are stored The parameter is name of an existing directory The return value is TRUE if successful

Method	Parameter or return value	Description
BOOL FindFirst(LPCTSTR lpszPattern);	Parameter = LPCTSTR lpszPattern Return value = BOOL	Finds the first pcAnywhere host object file (*.bhf) in the current directory, based on the specified file name pattern The parameter is the File name pattern to filter object files (an asterisk [*] finds all host files in the current directory) The return value is 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 FindFirst(BSTR FAR* pbstrFullQualName);	Parameter = BSTR FAR * pbstrFullQualName	Return buffer for the full path name of the remote object file that matches the specified pattern The return value is 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);	Parameter = BSTR FAR * pbstrFullQualName Return value = BOOL	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 The parameter is the return buffer for the full path name of the host object file that matches the pattern specified in the original call to FindFirst() The return value is 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);	Parameter = LPCTSTR lpszFQName	Retrieves a CHostData object by file name. The parameter is the fully qualified host object file name to be loaded

Method	Parameter or return value	Description
LPDISPATCH RetrieveObject(short wAccessMode);	Parameter = short wAccessMode	Specifies how this object is to be used. This relates to the password protection. The options are as follows: ■ 0 = Not specified ■ 1 = View only ■ 2 = View and Modify ■ 3 = Execute
LPDISPATCH RetrieveObject(LPCTSTR lpszPassword);	Parameter = LPCTSTR lpszPassword	Object password. May be NULL
LPDISPATCH RetrieveObject	Return value = LPDISPATCH	Pointer to an OLE dispatch object. The object is a CHostData object. See Visual C++ sample code for host functionality .
LPDISPATCH RetrieveObjectEx(LPCTSTR lpszFQName);	Parameter = LPCTSTR lpszFQName	Retrieves a CHostDataEx object by file name The parameter is the fully qualified host object file name to be loaded
LPDISPATCH RetrieveObjectEx(short wAccessMode);	Parameter = short wAccessMode	Specifies how this object is to be used. This relates to the password protection. The options are as follows: ■ 0 = Not specified ■ 1 = View only ■ 2 = View and Modify ■ 3 = Execute
LPDISPATCH RetrieveObjectEx(LPCTSTR lpszPassword);	Parameter = LPCTSTR lpszPassword	Object password. May be NULL
LPDISPATCH RetrieveObjectEx	Return value = LPDISPATCH	Pointer to an OLE dispatch object. The object is a CHostDataEx object See Visual C++ sample code for host functionality .

Method	Parameter or return value	Description
LPDISPATCH CreateObject(LPCTSTR lpszName);	Parameter = LPCTSTR lpszFQName Return value = LPDISPATCH	Creates a CHostData object and returns an LPDISPATCH pointer to it The parameter is the fully qualified host object file name for the new object The return value is a pointer to an OLE dispatch object. The object is a CHostData object See Visual C++ sample code for host functionality .
LPDISPATCH CreateObjectEx(LPCTSTR lpszName);	Parameter = LPCTSTR lpszFQName Return value = LPDISPATCH	Creates a CHostDataEx object and returns an LPDISPATCH pointer to it The parameter is the fully qualified host object file name for the new object The return value is a pointer to an OLE dispatch object. The object is a CHostDataEx object. See Visual C++ sample code for host functionality .
BOOL DeleteObject(LPCTSTR lpszFQName);	Parameter = LPCTSTR lpszFQName	Deletes a host object file The parameter is the fully qualified host object file name of the object to be deleted
BOOL DeleteObject(LPCTSTR lpszPassword);	Parameter = LPCTSTR lpszPassword	Object password. May be NULL
BOOL DeleteObject	Return value = BOOL	TRUE if object is deleted
BOOL Launch(LPCTSTR lpszFQName);	Parameter = LPCTSTR lpszFQName Return value = BOOL	Launches a host object file, which opens the pcAnywhere host terminal window The parameter is the fully qualified host object file of the object to be launched The return value is 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 whether 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
- 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
- ISDN via CAPI 2.0
- 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 are as follows:

- None
- Odd
- Even
- Mark
- Space

The following code sets the flow control levels:

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

Flow control values are as follows:

- <None>
- XONXOFF
- RTS/CTS

- BOTH

The following code sets the connection start values:

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

Connection start values are as follows:

- 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 are as follows:

- 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 are as follows:

- 110
- 300
- 600
- 1200
- 2400

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

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

The following table describes the normal methods of the object (they are not used to get and set properties):

Method	Parameter or return value	Description
short ConnectionTypes();	Return value = Short	Returns the number of connection types available on this computer
BSTR FirstConnectionType(); and BSTR NextConnectionType();	Return value = BSTR	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. You can use these returned connection types with the SetConnectionType() function The return value is the name of a supported connection device type
BOOL FindConnectionType(LPCTSTR lpszConnectionType);	Parameter = LPCTSTR lpszConnectionType Return value = BOOL	The parameter is the name of a connection device type The return value is TRUE if this device type is available

Method	Parameter or return value	Description
short MaxAssignedConnections()	Return value = Short	Returns the maximum number of connection types that can be assigned at the same time (currently two)
short AssignedConnections()	Return value = Short	Returns the number of assigned connection types.
BSTR FirstAssignedConnection(); and BSTR NextAssignedConnection();	Return value = BSTR	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. You can use these returned connection types with the AssignConnection() function The return value is the name of a supported connection device type
BOOL FindAssignedConnection (LPCTSTR lpszConnectionType);	Parameter = LPCTSTR lpszConnectionTyp Return value = BOOL	The parameter is the name of a connection device type The return value is TRUE if this device type is currently assigned
short CountryCodes();	Return value = Short	Returns the number of country codes available
BSTR FirstCountryCode(); and BSTR NextCountryCode();	Return value = BSTR	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. You can use these returned values with the SetCountryCode() function. The return value is the first or next country code string
BOOL ReadObject(LPCTSTR lpszPassword);	Parameter = LPCTSTR lpszPassword Return value = BOOL	Reads the object data from the host object file The parameter is the object password The return value is TRUE if object is successfully read

Method	Parameter or return value	Description
BOOL WriteObject(LPCTSTR lpszPassword);	Parameter = LPCTSTR lpszPassword Return value = BOOL	Writes the object data out to the host object file The parameter is the object password The return value is 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);
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, and 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 following table describes the Awrem32 functions, including the parameters and return values for handling connections between a host and remote computer:

Method	Parameter or return value	Description
boolean awConnect(BSTR FileName);	Parameter = Name as string Return value = Boolean	Creates the connection to the host computer The parameter is the fully qualified .chf file name that contains information about the host computer The return value executes the command
boolean awDisconnect();	Return value = Boolean	Disconnects the host computer After calling this function, the calling program must delete the object (C++ - delete IAwrem32X*, VB - set ObjectName = Nothing;)
boolean FileXferFromHost(BSTR HostFile);	Parameter = HostFile as string	Copies a file from the host computer to the remote computer. The parameters can contain wildcard characters The parameter contains the fully qualified path and file name to be copied from the host computer. The HostFile and RemoteFile strings do not have to be identical
boolean FileXferFromHost(BSTR RemoteFile);	Parameter = RemoteFile as string	Copies a file from the host computer to the remote computer. The parameters can contain wildcard characters The parameter contains the fully qualified destination path and file name. The HostFile and RemoteFile strings do not have to be identical.
boolean FileXferFromHost	Return value = Boolean	TRUE if command executed

Method	Parameter or return value	Description
boolean FileXferToHost(BSTR HostFile);	Parameter = HostFile as string	Copies a file from the remote computer to the host computer. The parameters can contain wildcard characters The parameter contains the fully qualified destination path and file name. The HostFile and RemoteFile strings do not have to be identical
boolean FileXferToHost(BSTR RemoteFile);	Parameter = RemoteFile as string	Copies a file from the remote computer to the host computer. The parameters can contain wildcard characters The parameter 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
boolean FileXferToHost	Return value = Boolean	TRUE if command executed
boolean CreateFolderOnHost(BSTR FolderName);	Parameter = FolderName as string Return value = Boolean	Creates a new folder on the host computer. This function creates a temporary folder on the remote computer, and then copies that folder to the host computer The parameter contains the drive and path to create the folder on the host computer The return value is TRUE if command executed

Method	Parameter or return value	Description
boolean ExecuteHostFile(BSTR FileName);	Parameter =FileName as string Return value = Boolean	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 The parameter contains the fully qualified path to the file on the host computer The return value is TRUE if command executed
BSTR GetError();	Return value = String	Returns the last error generated in Awrem32 as a string
short ConnectionStatus();	Return value = Short	Returns the current status of your connection to the host computer The possible values include the following: ■ -1 = Lost connection ■ 0 = No connection ■ 1 = Session connected

Index

A

- API libraries 13
- automation controllers
 - about Visual Basic 12
 - about Visual C++ 13
- automation server.. *See* pcAnywhere Automation Server
- Awrem32
 - functions 55, 85

C

- C++.. *See* Visual C++
- CAPI connections
 - host properties 46, 80
 - remote properties 30, 68
- CHostData
 - using Visual Basic 36
 - using Visual C++ 75
- CHostDataEx
 - using Visual Basic 47
 - using Visual C++ 82
- CHostDataManager
 - using Visual Basic 34
 - using Visual C++ 71
- class definitions
 - importing 14
 - viewing 14
- code samples
 - Visual Basic 32, 54
 - Visual C++ 70, 84
- COM devices
 - host device details 42, 76
 - remote device details 26, 66
- connection types
 - assigning 41
 - host properties 38, 76, 80
 - remote properties 23, 66, 68
 - unassigning 41
- connections
 - ending 56, 85
 - returning errors 87

- connections *(continued)*
 - returning status 57, 87
 - starting 36, 55, 63, 85
- country codes
 - on host 81
 - on remote 69
- CRemoteData
 - using Visual Basic 21
 - using Visual C++ 63
- CRemoteDataEx
 - using Visual Basic 31
 - using Visual C++ 70
- CRemoteDataManager
 - using Visual Basic 18
 - using Visual C++ 60

D

- devices.. *See* connection types

E

- encryption settings
 - on host 47, 82
 - on remote 31–32, 70
- errors
 - returning 57, 87

F

- file transfer
 - from host 56, 85
 - to host 56, 86
- files
 - running on host 56, 87
- flow control 27, 43, 66, 77
- folders
 - creating on host 56, 86

G

- GUIDs 11

H

host objects
 creating 74
 deleting 36, 74
 device details 76
 dialing properties 42
 directories 34, 71
 finding 35–36, 72–73
 methods 80
 passwords 81
 starting 36, 74

I

identifiers. *See* GUIDs

ISDN

 CAPI connections
 host properties 46, 80
 remote properties 30, 68

M

modem connections

 on host
 COM properties 42, 76
 country codes 81
 dialing properties 42
 TAPI devices 77
 on remote
 COM properties 26, 66
 country codes 69
 dialing properties 25
 TAPI devices 66

N

NASI devices 79
 NCSI devices 79
 NetBIOS connections
 host properties 45, 79
 remote properties 29, 68
 network connections
 on host
 NetBIOS properties 45, 79
 on remote
 NetBIOS properties 29, 68

O

OLE Automation.. *See* pcAnywhere Automation Server

P

parity 27, 43, 66, 77
 passwords
 on host objects 81
 on remote objects 69
 pcAnywhere Automation Server
 about 10
 accessing with Visual Basic 12
 accessing with Visual C++ 13
 example uses 10
 registering GUIDs 11
 type libraries 13

R

remote engine
 automatically registering 11
 manually registering 12
 remote objects
 creating 62
 deleting 63
 device details 66
 dialing properties 25
 directories 18, 60
 files 19–20, 34, 61–62
 methods 68
 passwords 69

S

status
 returning 57, 87

T

TAPI devices
 on host 77
 on remote 66
 type libraries 13

V

Visual Basic
 accessing pcAnywhere Automation Server 12
 Awrem32 functions 55
 CHostData object 36
 CHostDataEx object 47
 CHostDataManager object 34
 code samples 32, 54
 CRemoteData object 21
 CRemoteDataEx object 31

Visual Basic (*continued*)

- CRemoteDataManager object 18

Visual C++

- accessing pcAnywhere Automation Server 13

- Awrem32 functions 85

- CHostData object 75

- CHostDataEx object 82

- CHostDataManager object 71

- code samples 70, 84

- CRemoteData object 63

- CRemoteDataEx object 70

- CRemoteDataManager object 60

- importing classes 14

- including Winawsvr.h 15

W**Winawsvr**

- executable 12

- header file 15

- objects

 - CHostData 36, 75

 - CHostDataEx 47, 82

 - CHostDataManager 34, 71

 - CRemoteData 21, 63

 - CRemoteDataEx 31, 70

 - CRemoteDataManager 18, 60